DECTRIS®

Socket Interface Specification

MYTHEN Detector System

Version 2.1.0

DECTRIS®

Table of Contents

DECTRIS®

1 Document History

1.1 Actual document

Version	Date	Status	Prepared	Checked	released
2.1.0	02.07.2013	released	PetT	ChrH	ChrH

1.2 Changes

Version	Date	Changes
1.0	01.07.2009	First version
1.1	26.11.2009	Introduction, Status command
1.2	21.12.2009	Unix example application
1.3	07.04.2010 25.05.2010	New command to set dynamic range, version command, auto-readout command Bug fix: -read does not take an argument
2.0.0	23.12.2010	Complete revision of the socket server
2.0.1	24.03.2011	Bug fixes in flatfield correction and readout of bad channels
2.0.2	28.03.2011	Reset command must not change number of active modules
2.0.3	07.06.2011	Server compatibility with optimised highgain settings
2.0.4	16.11.2011	Bug fixes and updates for Mythen24K, new commands to get sensor material, sensor thickness and system serial number, improved initialisation of the number of modules
2.0.5	09.12.2011	New command to get the assembly date
2.0.5	05.03.2012	Layout adaptions Improvements to the section *Initialization*.
2.0.5	18.12.2012	Added *Known Issues* section to the release notes of version 2.0.5.
2.1.0	02.07.2013	Bad channel interpolation for clusters, dead time (tau) per module, apply rate correction before flatfield correction, bug fix for increasing number of modules

2 How to use this documentation

Before you start to operate the MYTHEN detector system please read the User Manual and the Technical Documentation included in the documentation package carefully.

This document has been designed for the MYTHEN detector systems.

2.1 Address and Support

DECTRIS Ltd.
Täfernweg 1
5405 Baden-Dättwil
Switzerland
Phone: +41 56 500 21 02
Fax: + 41 56 500 21 01

Website:

- www.dectris.com → support → Technical Notes → MYTHEN
- www.dectris.com → support → FAQ
- www.dectris.com → support → Problem Report

Email:

- support@dectris.com

Should you have questions concerning the system or its use, please contact us via phone, mail or fax.

	**Before you ship the system back, please contact DECTRIS Ltd. to receive the necessary transport and shipping information.**

	**If there is a cross-reference to other documents, "Vx_y"refers to the current version.**

2.2 Explanation of Symbols

Symbol	Description
	Important or helpful notice
	Caution. Please follow the instructions carefully to prevent equipment damage or personal injury.
	DC-current
	AC-current
	Ground
	Functional earth

2.3 Explanation of Terms

Term	Description
GUI	**G**raphical **U**ser **I**nterface
DCS1	**D**etector **C**ontrol **S**ystem for 1 detector module
DCS6	**D**etector **C**ontrol **S**ystem for up to 6 detector modules
DCS24	**D**etector **C**ontrol **S**ystem for up to 24 detector modules
Detector module	The smallest fully functional unit of the detector (1280 channels).
FIFO	Storage working according to the **F**irst **I**n - **F**irst **O**ut principle
keV	**K**ilo **e**lectron **V**olt

2.4 Disclaimer

DECTRIS Ltd. has carefully compiled the contents on this manual according to the current state of knowledge. Damage and warranty claims arising from missing or incorrect data are excluded.

DECTRIS Ltd. bears no responsibility or liability for damage of any kind, also for indirect or consequential damage resulting from the use of this system.

DECTRIS Ltd. is the sole owner of all user rights related to the contents of the manual (in particular information, images or materials), unless otherwise indicated. Without the written permission of DECTRIS Ltd. it is prohibited to integrate the protected contents published in these applications into other programs or other Web sites or to use them by any other means.

DECTRIS Ltd. reserves the right, at its own discretion and without liability or prior notice, to modify and/or discontinue this application in whole or in part at any time, and is not obliged to update the contents of the manual.

3 Warnings

	**Please read these warnings before operating the detector system.**

- DO NOT TOUCH THE ENTRANCE WINDOW OF THE DETECTOR.
- Place the protective cover on the entrance window of the detector when it is not in use.
- The detector is not specified to withstand direct beam at a synchrotron. Such exposure will damage the exposed channels.
- The detector system should have enough space for proper ventilation. Operating the detector outside the specified ambient conditions could damage the system.
- The air inlets and outlets of the detector control system should not be blocked.
- Power down the detector system before connecting or disconnecting any cable.
- Before connecting the power supply to the mains-supply, check the supply voltage with the label on the power supply. Using an improper main voltage will destroy the power supply and could damage the detector.
- Make sure the cables are connected and properly secured.
- Avoid pressure or tension on the cables.
- Opening the detector, the detector control system or the power supply housing without explicit instructions from DECTRIS Ltd. will void the warranty.
- The embedded Linux operating system on the detector control system has customized software for controlling the MYTHEN detector system. Do not make any changes to the Linux operating system without explicit instructions from DECTRIS Ltd..

4 Introduction

The DCS hosts an embedded Linux system, on which the Mythen Socket Server is running. A client can open a socket connection to the server and control the Mythen detector by sending the commands listed in chapter 7. Example applications for Linux and Windows are provided on the system CD. A *spec* implementation based on this socket interface is available from http://www.certif.com.

5 Operation

5.1 Connecting to the server

As a first step, the client has to open a socket connection to the server. The user can choose whether he wants to use UDP or TCP as protocol. While TCP is more reliable, UDP is faster and allows for higher frame rates.

To open the connection, the user has to specify the IP of the DCS, which is by default 192.168.0.90. The port of the socket server is 1030 for the UDP and 1031 for the TCP protocol.

The commands sent to the server are ASCII strings (i.e. character arrays). The response is also a character array, which has to be interpreted as string, integer array oder float array, depending on the command. All commands return responses of well defined lengths. Since the response might be fragmented on its way through the network, the client might have to read several packages, till the expected response length is reached.

In case of an error, the server will respond with a negative number. The meaning of the error codes can be found in section 7.1.

5.2 Initialization

First of all it should be checked, that the socket server has the expected version. This is done with help of the command "-get version".

Afterwards the detector system should be initialized by invoking the "-reset" command, which takes about two seconds per module. This command loads standard settings for Cu-radiation for all modules and enables the flatfield correction and the interpolation for bad channels (more details on the default values after initialisation can be found in section 7).

For proper operation all delivered modules have to be connected to the DCS. Nonetheless it is possible to read out only the first few modules. The number of active modules to be used can be specified using the “-nmodules” command.

To check, whether the communication with all modules is working, the module number for each module can be read out ("-get modnum"). Before reading a module number, the corresponding module has to be selected with the "-

module" command. Please note, that all module specific commands (e.g. the commands to load some settings) only affect the currently selected module. After initialisation the module at position 0 is selected. When working with a single module system the user does not need to take care about selecting modules.

5.3 Settings

There are different ways to choose the settings for the Mythen system. Depending on the calibration of your system, *fast* settings might not be available.

5.3.1 Predefined Settings

The predefined settings can be loaded with help of the "-settings" command. As an example "-settings StdCu" will load standard settings for Cu-radiation. The threshold subsequently can be modified with the "-kthresh" command. The more the threshold deviates from the original value for the loaded settings, the more the Mythen performance degrades with respect to the homogeneity of the threshold and sensitivity over the modules.

5.3.2 Automatic Settings

The "-autosettings" commands lets the user specify a threshold and tries to load optimal settings for the specified value.

5.3.3 High frame rates

To maximise the frame rate, the user should consider the following options:

1. Minimising the number of bits which are read out with the "-nbits" command.
2. Disabling the server side rate correction with the command "-ratecorrection 0".
3. Disabling the server side flatfield correction with the command "-flatfieldcorrection 0".
4. Using a UDP connection instead of the slower TCP connection.
5. Using the -readoutraw command instead of the slower -readout command.

5.4 Acquisition control

Before starting an acquisition, the number of frames and the duration of the frames have to be programmed. The former can be done with the "-frames" command, the latter with the "-time" command. Since the duration is specified in units of 100ns, the argument can be a large number. Therefore the user should make sure to handle these numbers correctly. For example in the C programming language, it will be necessary to use the data type *long long*, which comprises 64 bits.

An acquisition is started with the "-start" command and subsequently read out with the "-readout" command. A readout command that is sent during the acquisition will wait till the acquisition has finished and then return the result. An acquisition can be interrupted with the "-stop" command, which results in the loss of the result.

The status of the acquisition can be checked with the "-status" command. Sending commands other than "-readout", "-status" or "-stop" during an acquisition can result in misbehaviour of the detector.

5.5 Data corrections

For channels marked as bad the server will replace the number of counts with the average number of counts of the next lower and the next upper working channels. With the "-badchannelinterpolation" command the user can turn off this correction. In this case the number of counts will equal to -2 for these channels.

The server automatically applies a flat-field correction to the data. This behaviour can be disabled with the "-flatfieldcorrection" command.

For high incoming x-ray fluxes, the rate correction can be turned on with the "-ratecorrection" command.

5.6 Triggering and gating

The standard and the continuos trigger modes can be enabled by the "-trigen" and "-conttrigen" commands. The timing of the measurements can be adjusted with the "-delbef" and "-delafter" commands.

The gating mode is turned on by the "-gateen" command. The number of gates can be programmed with the "-gates" command.

The polarity of the input and output signals can be defined with the help of the "-inpol" and "-outpol" commands.

5.7 Debugging

After sending the command "-log start" the socket server will log his activities into a file on the DCS until a "-log stop" command is sent. The content of the file is sent to the user in response to the "-log read" command. Since there is only limited disk and memory space on the DCS, the logging functionality should only be turned on for short periods of time.

For debugging problems related to the transmission of the data packages over the socket, the freely available network protocol analyser WireShark can be used.

6 Example Applications

As a starting point a Unix and a Windows C application with some comments are provided on the system CD in the folder software/doc. They set up a socket connection to the DCS and read back the module serial number. Afterwards a short acquisition is performed, whose result is printed on the command prompt. The Windows version was developed with the freely available Microsoft Visual C++ 2008 Express Edition.

7 Commands

Synopsis: Specifies the format of the command. Values in angle brackets have to be replaced by the user: *b* stands for a boolean (i.e. a 0 or 1), *n* for an integer, *f* for a float, *time* for a 64-bit integer (long long or int64_t , used for the timing commands).
Arguments: Gives the meaning of the arguments. Default values after initialisation are set in *italic* type.
Description: Describes the effect of the command.
Return Type: Specifies the data type of the returned byte array. If no return type is indicated, the command returns an integer: a zero indicating successful execution of the command or a negative number in case of an error (see section Error Codes). If the return value is of type float, the error code is returned as a float number, and as an integer otherwise.
Return Size: Specifies the number of data elements in the returned byte array. For example a response of return type *int* and return size 1280 is 5120 bytes long (since one integer uses 4 bytes). N_{MOD} stands for the number of active modules (set by "-nmodules").

Synopsis	Argument	Description	Return Type	Return Size
-autosettings <f>	f > 0 in units of keV	Sets the threshold to the specified value and loads suitable settings for the current module. If the threshold is below 8.05 keV, highgain settings and the Cr flatfield are used. Above this value, standard settings and the Mo flatfield are loaded. The trim bits are loaded from the calibration, which differs least in energy from the specified threshold. This command takes about two seconds per module.		
-badchannelinterpolation <b>	0: Disable *1: Enable*	Turns on or off the interpolation routine for bad channels. When enabled, the number of counts for a bad channel is set to the average number of counts of the next lower and the next upper		

		working channels. If there is no next lower (upper) working channel, the number of counts of the next upper (lower) working channel is used. When disabled the number of counts for a bad channel is set to -2.		
-conttrigen <b>	*0: Disable* 1: Enable	Enables or disables the continuos trigger mode. In this mode, each frame requires a trigger signal.		
-delbef <time>	Delay in units of 100ns. Default is *0ns*.	Sets the delay between a trigger signal and the start of the measurement.		
-delafter <time>	Delay in units of 100ns. Default is *0ns*.	Sets the delay between two subsequent frames.		
-flatfieldcorrection <b>	0: Disable *1: Enable*	Enables or disables the flatfield correction. After initialisation the flatfield correction is enabled.		
-frames <n>	Number of frames. Default is *1*.	Sets the number of frames within an acquisition.		
-gateen <b>	*0: Disable* 1: Enable	Enables or disables the gated measurement mode.		
-gates <n>	Number of gates. Default is *1*.	Sets the number of gate signals within one frame.		
-get assemblydate		Returns the assembly date of the system as character array.	char	50
-get badchannelinterpolation		Returns whether the bad channel interpolation is enabled (1) or not (0).	int	1
-get badchannels		Returns the status of each channel. A value of 0 means that the channel is working, a value of 1 means that it is defective.	int	N_{MOD}*1280
-get flatfield		Returns the currently loaded flatfield calibration of all modules.	int	N_{MOD}*1280

-get flatfieldcorrection		Returns whether the flatfield correction is enabled (1) or not (0).	int	1
-get kthresh		Returns the threshold of the current module in keV	float	1
-get modnum		Return the serial number of the current module, which has to be interpreted as a hex-number.	int	1
-get nbits		Returns the number of bits to be read out.	int	1
-get nmodules		Returns the number of active modules	int	1
-get ratecorrection		Returns whether the rate correction is enabled (1) or not (0).	int	1
-get sensormaterial		Returns an identifier for the sensor material 0: Silicon	int	1
-get sensorthickness		Returns the nominal sensor thickness in µm.	int	1
-get settings		Returns the settings of the current module 0: Standard, 1: Highgain, 2: Fast, 3: Unknown	int	1
-get settingsmode		Returns the last loaded settings for the current module. If predefined settings were loaded, the returned string is of the form "predefined StdCu". If the autosettings command was used, it is of the form "auto <thr>", with <thr> being the threshold supplied with the autosettings command.	char	50
-get status		Returns the status of the DCS as a bit pattern: Bit 0: Run busy Bit 3: Waiting for trigger Bit 16: All FIFO empty	int	1
-get systemnum		Returns the serial number of the Mythen system.	Int	1
-get tau		Returns the programmed dead time constant for the current module in units of ns.	float	1
-get time		Returns the programmed exposure time in units of 100ns.	long long	1
-get version		Returns the software version of the socket server, which is of the form "M2.0.0". The returned string is terminated by a Null	char	7

		character.		
-inpol <b>	*0: rising edge / active high* 1: falling edge / active low	Sets the input polarity for trigger (rising or falling edge) and gate signals (active high or low).		
-kthresh <f>	f > 0 in units of keV	Sets the energy threshold of the current module.		
-log start		Tells the socket server to log his activities into a file on the DCS. Do not enable logging except for debugging purposes, since it will slow down the server and because there is only limited space for the log file on the DCS.		
-log stop		Stops the logging functionality of the socket server. The command returns the size of the log file.	int	1
-log read		Tells the socket server to send back the content of the log file. The length of the server response has to be inferred from the return value of the "-log stop" command.	char	
-nbits <n>	n = 4 | 8 | 16 | *24*	Sets the number of bits to be read out, therebye determining the dynamic range and the maximum frame rate. After initialisation 24 bits are read out.		
-module <n>	Module position $0 \leq n < N_{MOD}$	Selects the module at the given position as the target of module specific commands. After initialisation the module at position 0 is selected.		
-nmodules <n>	Number of modules	Sets the number of active modules (N_{MOD}). When the number of active modules is increased, proper settings have to be loaded for the additional modules. After initialisation the number of active modules is set to the number of delivered modules.		
-outpol <b>	*0: active high* 1: active low	Sets the polarity of the enable out signal.		

-ratecorrection <b>	*0: Disable* 1: Enable	Enables or disables the rate correction. After initialisation the rate correction is disabled.		
-readout		Returns the oldest data set in the FIFO. If the FIFO is empty and there is an ongoing measurement, the command returns the data after the measurement has finished. If the readout fails for some reason, all count values are set to -1.	int	N_{MOD}*1280
-readoutraw		Returns the oldest data set in the FIFO in a raw, compressed format. The DCS does not apply any correction to the data, thereby allowing for maximal frame rate. The example code in section 7.2 shows how to decode the raw data.	int	N_B*N_{MOD} N_{BITS} | N_B 24 | 1280 16 | 640 8 | 320 4 | 160
-reset		Sets the detector back to default settings. This command takes about two seconds per module.		
-settings <name>	name = *StdCu* | StdMo | HgCr | HgCu | FastCu | FastMo	Loads predefined settings for the current module to measure some common x-ray radiation. Std stands for standard, Hg for highgain, Fast for fast settings. Cu, Cr and Mo refer to the (fluorescence) energy of the incoming x-ray radiation. The command loads the energy calibration, the bad channels file, the flatfield correction, and the trimbits for the corresponding settings, and sets the energy threshold to a suitable value. This		

		command takes about two seconds per module.		
-start		Starts an acquisition with the programmed number of frames or gates.[1]		
-stop		Stops the current acquisition. The data of the ongoing frame is discarded.		
-tau <f>	f > 0 or f = -1.0 in units of ns	Sets the dead time constant used by the rate correction for the current module. If the argument is equal to -1.0, the default value for the active settings is going to be used.		
-time <time>	Exposure time in units of 100ns. Default is *1s*.	Sets the exposure time of one frame.		
-trigen <b>	*0: Disable* 1: Enable	Enables or disables the trigger mode. The acquisition starts after the trigger signal. Subsequent frames are started automatically.		

[1] In the legacy mode the first frame is read out automatically if the acquisition time is below the automatic readout time. In this case the return type and size are as stated for the readout command.

Deprecated commands

Synopsis	Argument	Description	Return Type	Return Size
-autoreadouttime <time>	time > 0 or time = -1 in units of 100ns	Sets the automatic readout time. If the acquisition time is below the automatic readout time, the first frame is read out automatically after the "-start" command. The automatic readout mechanism is disabled if the argument is equal to -1.		
-flipchannels <b>	0: Disable 1: Enable	Enables or disables the flipping of the channel numbering. In versions older than 2.0 the numbering was flipped compared to the MythenGUI. After initialisation the flipping of the channels is turned off.		
-get autoreadouttime		Returns the programmed automatic readout time in units of 100ns.	long long	1
-get flipchannels		Returns whether the channel flipping is enabled (1) or not (0).	int	1
-legacy		Enables the legacy mode for backwards compatibility with socket servers below version 2. This means that the auto readout time is set to 2s, flatfield correction is turned off and channel flipping is turned on.		
-read		Deprecated version of the "-get flatfield" command.	int	N_{MOD}*1280
-trimfile <name>	name = TrimCr | TrimCu | TrimMo	Deprecated version of the "-settings" command. The arguments TrimCr, TrimCu, TrimMo corresponde to the arguments HgCr, StdCu, StdMo of the "-settings" command in this order.		

7.1 Error Codes

If the response of the server consists of a single, negative number, an error has occured during the execution of the command. The following table lists the correspoding error codes.

Error code	*Description*
-1	Unknown command
-2	Invalid argument
-3	Unknown settings
-4	Out of memory
-5	Module calibration files not found
-6	Readout failed
-10	Flatfield not found
-11	Bad channel file not found
-12	Energy calibration not found
-13	Noise file not found
-14	Trimbit file not found
-15	Invalid format of the flatfield file
-16	Invalid format of the bad channel file
-17	Invalid format of the energy calibration file
-18	Invalid format of the noise file
-19	Invalid format of the trimbit file
-20	Version file not found
-21	Invalid format of the version file
-30	Could not create log file
-31	Could not close log file
-32	Could not read log file

7.2 Decoding the raw data

The example code below shows how to decode the compressed raw data returned by the "-readoutraw" command. The usage of this command is only necessary, when very high frame rates (> 300 Hz) are required. All corrections (flatfield-, rate-, bad channel-corrections) have to be applied afterwards by the client.

```
// Input
// nmods: number of active modules
// nbits: number of bits, which were read out
// data: response of the -readoutraw command
//
// Output
// result: array of size 1280*nmods with the number of counts of all
//         channels

void decodeRawReadout(int nmods, int nbits, int *data, int *result)
{
  int chanperline = 1;     // default for 24 bits
  int mask=0xffffff;       // default for 24 bits
  if (nbits == 16)
  {
    chanperline = 2;
    mask=0xffff;
  }
  if (nbits == 8)
  {
    chanperline = 4;
    mask=0xff;
  }
  if (nbits == 4)
  {
    chanperline = 8;
    mask=0xf;
  }

  int size = 1280/chanperline*nmods;
  u_int32_t tmpArray[size];                  // the data has to interpreted
  memcpy(tmpArray, data, size*sizeof(int));  // unsigned int32

  for (int j = 0; j < chanperline; j++)
  {
    int shift = nbits*j;
    int shiftedMask = mask<<shift;
    for (int i = 0; i < size; i++)
    {
       result[i*chanperline+j]=((tmpArray[i]&shiftedMask)>>shift)&mask;
    }
  }
}
```

8 Release Notes

8.1 Version 2.1.0

- The bad channel interpolation now supports neighbouring bad channels. In this case the interpolated value is equal to the average number of counts of the next lower and the next upper working channels. If there is no next lower (upper) working channel, the number of counts of the next upper (lower) working channel is used.
- Different modules can now have different dead time constants for the rate correction. This fixes the first known issue of version 2.0.5.
- Increasing the number of active modules now works flawlessly, fixing the second known issue of version 2.0.5.
- The rate correction is now applied before the flatfield correction.

8.2 Version 2.0.5

New command to get the assembly date of the system („-get assemblydate“).

8.2.1 Known issues

- When operating a multi-module system with differing gain settings (Std, Hg, Fast) for the various modules, the rate correction uses the dead time (“tau”) constant of the currently selected module for all modules. Correctly it would use different dead time constants depending on the gain of each module. Therefore the rate correction should only be used when all modules of a multi-module system have the same gain settings.
- When setting the number of active modules to a value larger than the original number of delivered modules the server will stuck when loading settings for the additional modules. To correctly initialise additional modules, the serial number of the modules have to be added to the configuration file /etc/conf.d/version on the DCS.

8.3 Version 2.0.4

New commands (“-get sensorthickness”, “-get sensormaterial”, “-get systemnum”) to retrieve information about the sensor and the system. If the information can not be found, one of the new error codes -20 and -21 is returned. If you upgrade an older system to version 2.0.4, the sensor information will not be available.

Several improvments and bug fixes for multi-module systems (especially for the Mythen24K system) have been implemented:

- Faster execution of the reset command.
- After execution of the reset command the first module (module 0) is selected (instead of the last one).
- Correct sending of data if its length is a multiple of the UDP/TCP packet size.
- The initialisation sets the number of modules to the number of delivered modules. For systems, which are delivered with less modules than the DCS maximally supports (e.g. a Mythen3K with 3 modules on a DCS6), the user no longer needs to specify the number of modules. The command to set the number of active modules is only necessary for running a system with less (or more) than the number of delivered modules. In contrast to earlier versions (2.0.2 and 2.0.3), the corresponding command has to be executed after the reset command, since a reset now sets the number of modules to the number of delivered modules.

8.4 Version 2.0.3

The server is now able to handle optimised highgain settings. This change is purely internal and has no consequences for the user.

8.5 Version 2.0.2

The reset command does not change the number of active modules any more. This prevents the issuing of error messages by the reset command when working with a not fully equipped multi-module system.

8.6 Version 2.0.1

Two bugs in version 2.0.0 have been fixed:

- Due to a type conversion problem, the server-side flatfield correction truncated the correction factors to integers. As a consequence many channels returned zero counts in all readouts. The problem only occurred when the server-side flatfield correction was turned on.
- The result of the” -get badchannels” did not have the format as described in this specification.

8.7 Version 2.0.0

Version 2.0.0 is a complete redesign of the socket server and the user is encouraged to adapt his older implementation to the new version, since it offers better stability and more functionality.

8.7.1 New Functionalities

- The server can be controlled over a TCP socket
- Support for up to 24 modules
- Fast settings are available to the user
- Built-in support for flat-field correction
- Built-in support for rate correction
- Automatic loading of optimal settings based on the required threshold
- Many new commands to read out the current settings and parameters
- Logging functionality

8.7.2 Migration notes

If you migrate from version 1.3.0, take care of the following changes:

- By default the automatic readout of the first frame in an acquisition is disabled.
- The flipping of the channels is turned off after initialisation.
- By default the flat-field correction is performed directly on the DCS.
- The "-read" command is deprecated and should be replaced by the "-get flatfield" command.
- The "-trimfile" command is deprecated and should be replaced by the "-settings" command.

8.7.3 Legacy Mode

If for some reasons you are not going to migrate, you can enable the legacy mode with the "-legacy" command. After having sent this command, the server should behave identically to version 1.3.0 up to bug fixes and the more detailed error responses.