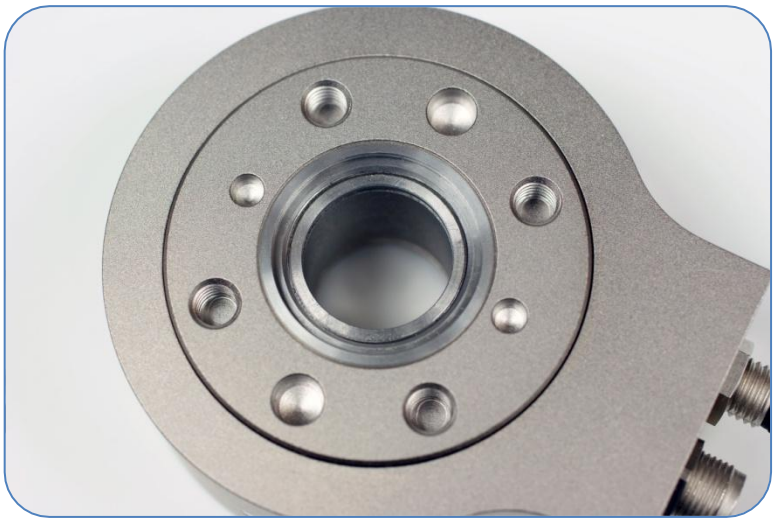
KMS Command Set

Reference Manual

Firmware Version 1.2.0

August 2014



Contents

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

1 Introduction

The KMS family of Force/Torque Sensors can be controlled by a number of interfaces using different command sets. While the CAN Bus interface uses a binary protocol adapted to the 8-byte CAN message format, the Ethernet based interface and the RS-232 serial line interface provide a text-based command set. This manual gives a detailed explanation of the text based command set used for Ethernet and serial line communication. For the CAN Bus interface, please refer to the "KMS CAN Bus Interface Manual".
To get started with the communication protocol, we recommend using a standard Telnet client like the free PuTTY[1] for Microsoft Windows.

1.1 Connecting to the KMS

The KMS family of Force/Torque Sensors offers a number of interfaces, each of which uses the same text-based message protocol as described in this manual. Available interfaces are Ethernet TCP/IP, Ethernet UDP/IP and RS-232 serial connection (optional).
The interface configuration can be changed using the sensor's web interface. Connect the KMS to the local network or directly to your computer's network interface and point your favorite web browser to the sensor's IP address, e.g. by typing the default **http://192.168.1.30** into the address bar and pressing "Enter". After the page has loaded, choose "*Settings -> Command Interface"* from the menu to configure the default interface.
Use the TCP interface to allow connections using a telnet client like PuTTY.

1.2 Communicating with the KMS

Regardless of the interface being used, the KMS communicates with its client using a text-based protocol. The following chapters describe the general format of these commands.
The KMS expects commands being submitted as plain ASCII strings. Each command must be terminated by a line feed character ('\n' or ASCII code 0x0d).

[1] http://www.putty.org/

Return messages are submitted in the same format. For details, check the command descriptions in chapter 2.
The sensor's command interface interprets the received command strings as chunks of code written in the LUA programming language[2]. This enables the use of several programming techniques like *for*-loops or conditional branches. An example of how to use these features is given in the sensor's Scripting Reference Manual.

1.3 Error handling

In case of an error, the sensor returns a message string of the following format:

ERROR(<integer>)

where <integer> represents one of the error codes described in chapter 3.
If the verbose level is increased to 1, the sensor submits extended error messages containing an additional text string that describes the type of error:

ERROR(<integer>, <string>)

☞ **See chapter 3 to get a description of the error codes.**

☞ **See chapter 2.1.1 on how to increase the verbose level.**

1.4 Connecting to the command interface using PuTTY

PuTTY is a free Telnet and SSH client that can be used to connect to the sensor's command interface. The following chapter shows how to use PuTTY with the KMS.
Before you start, please make sure the sensor is configured to use the TCP/IP interface and check its IP address and port. **The default IP address of the sensor is 192.168.1.30, default TCP port is 1000.**
Download and install PuTTY from **http://www.putty.org**.
After starting PuTTY, a new connection must be configured. Type in the IP address and port number of the sensor and set the connection type to "raw" (see fig. 1).
As the sensor does not send a carriage return character in its response messages, PuTTY must be configured to explicitly do a carriage return on each line feed character. In the settings window, open the category "Terminal" and enable "Implicit CR in every LF" (see fig. 2).

[2] http://lua.org

Now click the “Open” button to open the connection. A new and empty terminal window will appear (fig. 3), ready to type in your commands. The command syntax is explained in detail in chapter 2. Typing “F()” for example, followed by <Enter>, will return the currently measured values (see chapter 2.3.1).

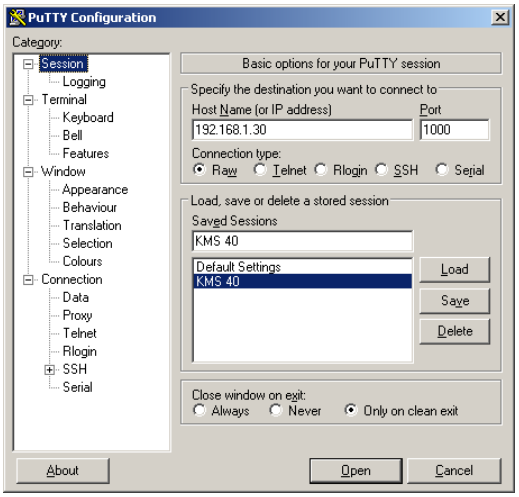


Figure 1: Session settings

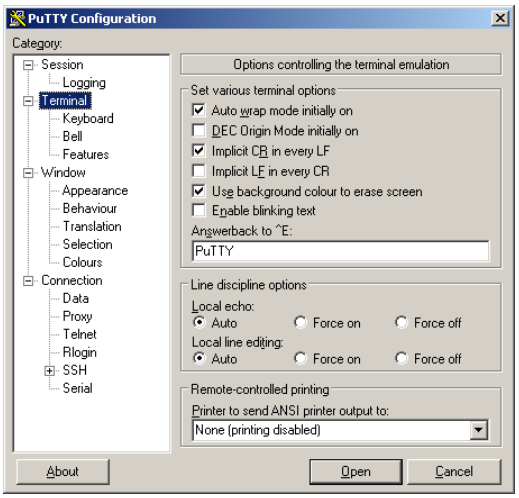


Figure 2: Terminal settings

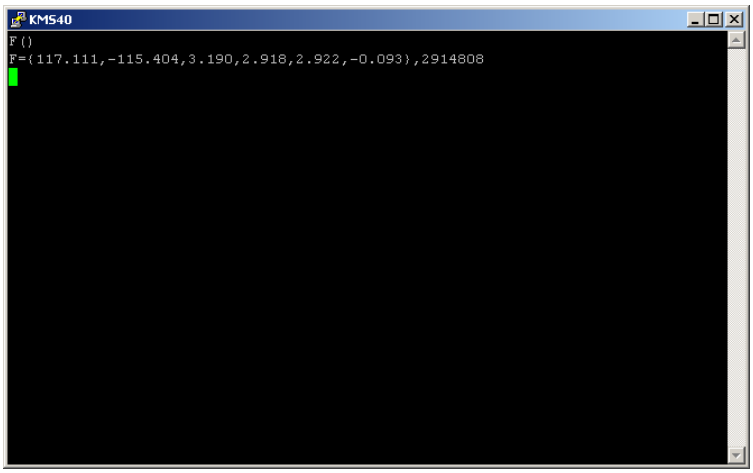


Figure 3: Terminal window

2 Command Set Reference

The following chapter describes the command set of the KMS in detail.

2.1 Interface Control

2.1.1 Get or set verbose level – VL()

Set verbose level of the interface. By default, the verbose level is set to 0, meaning that the sensor only returns an error code with its error messages. By increasing the verbose level to 1, the sensor also returns a text string describing the error.

Syntax

VL([<integer>])

Parameters

Integer representing the verbose level.

Return string

VL=<integer>
e.g. VL=0

2.1.2 Print variable – PRINT()

Output variables to the command interface. This command is intended to be used in conjunction with the interactive scripting extensions mentioned in chapter 1.2. Its only purpose is to print out the values of variables that have been assigned previously to the command interface.

☞ **Please refer to the Scripting Reference Manual for an example on how to use interactive scripting on the command interface.**

Syntax

PRINT(<var>, <var>, …)

Parameters

<var> A variable that has been assigned previously.
The command takes a variable number of arguments.

Return string

PRINT=<value>,<value>, …
The command prints out the values of the given variables.

2.2 System State

2.2.1 Get system type – ID()

This command returns the type of the system and can be used for example to distinguish between different devices manufactured by Weiss Robotics that support this protocol.

Syntax

ID()

Parameters

none

Return string

ID=<string>

The command returns the system type string, e.g. *ID="KMS 40"*.

2.2.2 Get firmware version – V()

Returns the firmware version of the sensor.

Syntax

V()

Parameters

none

Return string

V=<string>

The command returns the firmware version string, e.g. *V="1.0.0"*.

2.2.3 Get serial number – SN()

Returns the sensor's serial number.

Syntax

SN()

Parameters

none

Return string

SN=<integer>

The command returns the sensor's serial number, eg. *SN=12345678*.

2.2.4 Get or set descriptor string (device tag) – D()

This command can be used to read and/or write the sensor's descriptor string. The descriptor string, also known as device tag, can be used to identify the device.

Syntax

D([<string>])

Parameters

<string> *Optional tag string to be set*

Return string

D=<string>

The command returns the sensor's descriptor string, e.g. *D="myDescriptor"*.

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

2.2.5 Get system flags – FLAGS()

Get the system flags value. The returned value is the decimal representation of a bit vector of 32 bit length, encoded as an integer value. A return value of 12 (binary 1100), for example, indicates that bit 2 and bit 3 are set, which means that the flags SF_FILTER_EN and SF_TARA are enabled.
Chapter 4 gives an overview of the available flags.

Syntax

FLAGS()

Parameters

none

Return string

FLAGS=<integer>

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

2.2.6 Get temperature – T()

Get the current system temperature in degrees Celsius.

Syntax

T()

Parameters

none

Return string

T=<number>

in degrees Celsius, e.g. T=34.2

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

2.3 Data Acquisition

2.3.1 Acquire single frame – F()

Get a single data frame including a timestamp. The first three numbers in the return string indicate the forces F_x, F_y and F_z (in this order) in Newton (N). The three following numbers indicate the torque values M_x, M_y and M_z in Newton meters (Nm). The last number indicates the time stamp in 1/10 Milliseconds (ms).

Syntax

F()

Parameters

none

Return string

F={<float>,<float>,<float>,<float>,<float>,<float>},<integer>
e.g. F={20.123,-67.746,-0.439,-0.342,4.342,0.978},472416
The values in braces represent the measured data {F_x, F_y, F_z, M_x, M_y, M_z} in this order. The last value represents the time stamp.

2.3.2 Start continuous data acquisition – L1()

Start continuous acquisition of sensor data frames. The sensor will repeatedly submit frame data. The first 3 numbers in the return string indicate the forces F_x, F_y and F_z (in this order) in Newton (N). The 3 following numbers indicate the torque values M_x, M_y and M_z in Newton meters (Nm). The last number indicates the time stamp in 1/10 Milliseconds (ms).

By default, the sensor will submit frames at a fixed rate of 500 frames per second that can be influenced using the `LDIV()` command. See chapter 2.3.5.

By default, the sensor will submit all measured data (six values). However, unused values can be suppressed using the `LMASK()` command. See chapter 2.3.4 for details.

Syntax

L1()

Parameters

none

Return string

1. *L1* (to confirm command)
2. *F={<float>,<float>,<float>,<float>,<float>,<float>},<integer>*

   repeatedly in continuous frame rate. The values in braces represent the measured data {F_x, F_y, F_z, M_x, M_y, M_z} in this order, force values F in Newton (N), torque values M in Newton meters (Nm). The last value represents the time stamp in 1/10 Milliseconds (ms).

   If the `LMASK()` command is used, some of the force/torque values may be suppressed, resulting in a frame of less than six values.

2.3.3 Stop continuous data acquisition – L0()

Stop continuous acquisition of data frames.

Syntax

L0()

Parameters

none

Return string

L0 (to confirm command)

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

2.3.4 Get or set data acquisition mask – LMASK()

For some applications, not all of the sensor's measured values may be relevant. In this case, the unnecessary values can be suppressed by masking them out. This removes the value from the data frame, limiting the necessary data rate for the interface to the absolute minimum.

Please note that this command will not influence the acquisition of single data frames using the `F()` command (chapter 2.3.1).

Syntax

LMASK({[<bit>,<bit>,<bit>,<bit>,<bit>,<bit>]})

Parameters

Bit vector of six values (optional) that indicates which values of the data frame should be enabled and which not. If no parameter is given, the command will simply return the current mask vector.

Return string

LMASK={<bit>,<bit>,<bit>,<bit>,<bit>,<bit>}
A value of 1 indicates that the corresponding value in the data frame is enabled, 0 means it is disabled.

Example

LMASK({1,0,0,1,0,0})
Will activate F_x and M_x in the data frame, whereas F_y, F_z, M_y and M_z will be disabled. In this case, the data acquisition will return frames that consist of two values only, e.g. something like *F={20.123,-10.456},472416*.

2.3.5 Get or set frame send divider – LDIV()

When continuously acquiring sensor frame data using the `L1()` and `LO()` command, the sensor will send frames at the maximum frame rate. To slow down the frame rate, a divider value can be set that tells the sensor only to submit every n-th frame, e.g. every 2nd, 3rd or 4th frame, dividing the maximum frame rate by this value.

To reset the frame rate to its maximum value, set the frame send divider value to 1.

Syntax

LDIV([<integer>])

Parameters

<integer> Optional integer value representing the new divider.

Return string

LDIV=<integer>

Returns the currently set divider value.

2.3.6 Tare or untare sensor – TARE()

Tare or untare the sensor and/or return if it is currently tared. If taring is selected, the sensor will record a number of measured values over a short time period and calculate average values that will be subtracted as an offset value from all future measured values, making the currently measured value become 0. If untaring is selected, this offset value will be cleared.

Syntax

TARE([<bit>])

Parameters

<bit> Optional bit value indicating whether the sensor should be tared (1) or untared (0).

Return string

TARE=<bit>

Returns whether the sensor is currently tared (1) or untared (0).

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

2.3.7 Get or set filter – FLT()

Get or set the type of filter that should be used to filter the sensor signals.

☞ **See Appendix C for a detailed description of the available filters.**

Syntax

FLTSET([<integer>])

Parameters

<integer> Optional integer value representing the ID of the filter that should be used. A value of 0 means that filtering is turned off.

Return string

FLTSET=<integer>

Returns the ID of the filter currently set.

2.4 Calibration

2.4.1 Get calibration date and lifetime– CALDATE()

Get the sensor's calibration date and lifetime. The sensor calibration is only valid for a limited time. Therefore, the calibration date may be important to detect at which point the calibration will become invalid.

Syntax

CALDATE()

Parameters

none

Return string

CALDATE=<integer>,<integer>

Returns two integer values representing the calibration date as a UNIX-like timestamp[3] and the calibration lifetime.

[3] Number of seconds passed since Jan. 1, 1970 0:00 h

2.4.2 Get calibration matrix – CALMATRIX()

The command returns the sensor's calibration matrix. This matrix is used to transform raw sensor data into calibrated force/torque values.

Syntax

CALMATRIX()

Parameters

none

Return string

CALMATRIX={{<number>, … <number>},…,{<number>, …<number>}}
Returns a 6x6 matrix of numbers representing the sensor's calibration matrix.

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

3 Appendix A: Error Codes

Error code	Symbol name	Description
0	E_SUCCESS	No error occurred, operation was successful
1	E_NOT_AVAILABLE	Function or data is not available
2	E_NO_SENSOR	No measurement converter is connected
3	E_NOT_INITIALIZED	Device was not initialized
4	E_ALREADY_RUNNING	The data acquisition is already running
5	E_FEATURE_NOT_SUPPORTED	The requested feature is currently not available
6	E_INCONSISTENT_DATA	One or more parameters are inconsistent
7	E_TIMEOUT	Timeout error
8	E_READ_ERROR	Error while reading data
9	E_WRITE_ERROR	Error while writing data
10	E_INSUFFICIENT_RESOURCES	No more memory available
11	E_CHECKSUM_ERROR	Checksum error
12	E_NO_PARAM_EXPECTED	A Parameter was given, but none expected
13	E_NOT_ENOUGH_PARAMS	Not enough parameters to execute the command
14	E_CMD_UNKNOWN	Unknown command
15	E_CMD_FORMAT_ERROR	Command format error
16	E_ACCESS_DENIED	Access denied
17	E_ALREADY_OPEN	Interface is already open
18	E_CMD_FAILED	Error while executing a command
19	E_CMD_ABORTED	Command execution was aborted by the user
20	E_INVALID_HANDLE	Invalid handle
21	E_ NOT_FOUND	Device or file not found
22	E_ NOT_OPEN	Device or file not open

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

23	E_IO_ERROR	Input/Output Error
24	E_INVALID_PARAMETER	Wrong parameter
25	E_INDEX_OUT_OF_BOUNDS	Index out of bounds
26	E_CMD_PENDING	No error, but the command was not completed, yet. Another return message will follow including an error code, if the function was completed.
27	E_OVERRUN	Data overrun
28	RANGE_ERROR	Range error
29	E_AXIS_BLOCKED	Axis blocked
30	E_FILE_EXISTS	File already exists

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

4 Appendix B: System State Flags

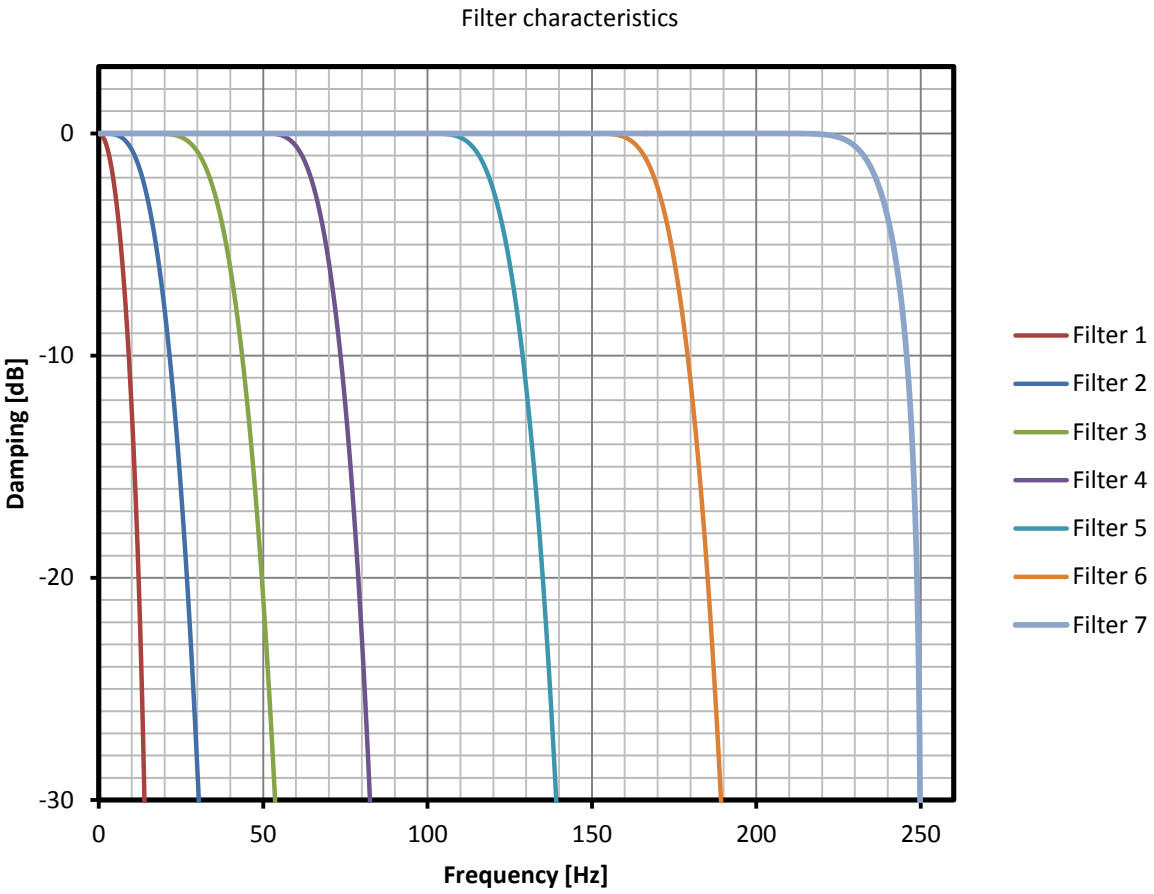
The system state flags are arranged as a 32-bit wide integer value that can be read using the command STATE() (see chapter 2.2.1). Each bit has a special meaning listed below.

Bit No.	Flag Name	Description
D31	reserved	These bits are currently unused but may be used in a future release of the KMS firmware.
D30	SF_SCRIPT_FAILURE	**Script Error.** An error occurred while executing a script and the script has been aborted. This flag is reset whenever a script is started.
D29	SF_CMD_FAILURE	**Command Error.** The last command returned an error.
D28	SF_POWER_FAULT	**Power Fault.** The power supply is out of range.
D27	SF_TEMP_FAULT	**Temperature Fault.** The sensor hardware has reached a critical temperature limit.
D26	SF_CAL_FAULT	**Calibration Fault.** Calibration fault.
D25	SF_OV_MZ	**Overrun M_z.** Overrun for M_z value.
D24	SF_OV_MY	**Overrun M_y.** Overrun for M_y value.
D23	SF_OV_MX	**Overrun M_x.** Overrun for M_x value.
D22	SF_OV_FZ	**Overrun F_z.** Overrun for F_z value.

D21	SF_OV_FY	**Overrun F_y.** Overrun for F_y value.
D20	SF_OV_FX	**Overrun F_x.** Overrun for F_x value.
D19..12	reserved	These bits are currently unused but may be used in a future release of the KMS firmware.
D11	SF_TEMP_WARNING	**Temperature Warning.** The sensor hardware will soon reach a critical temperature level.
D10	SF_CAL_EXPIRED	**Calibration expired.** Sensor calibration has expired
D9..6	reserved	These bits are currently unused but may be used in a future release of the KMS firmware.
D5	SF_SCRIPT_RUNNING	**Script running.** A script is currently running.
D4	SF_DAQ_RUNNING	**Data acquisition running.** Data acquisition is running. Frames are submitted using the command interface.
D3	SF_FILTER_EN	**Filtering enabled.** Filtering of sensor values is enabled.
D2	SF_TARA	**The sensor is tared.** Set if the sensor is tared.
D1	SF_STABLE	**Values stable.** The sensor measured values are stable.
D0	SF_CAL_VALID	**Calibration valid.** Sensor calibration is valid.

5 Appendix C: Filter types

The sensor offers seven predefined filter settings of different frequency range to filter the acquired data. The following diagram and table indicate the filter characteristics.



ID	Filter	Cutoff Frequency (-3 dB)
1	Filter 1	5 Hz
2	Filter 2	15 Hz
3	Filter 3	35 Hz
4	Filter 4	65 Hz
5	Filter 5	120 Hz
6	Filter 6	170 Hz
7	Filter 7	240 Hz

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany

Phone: +49 (0)7141 642062-6
Fax: +49 (0)7141 642062-9
email: office@weiss-robotics.com

6 Appendix D: Syntax Notation

The following command syntax notation is used throughout this document:

Parameters

a	Denotes a mandatory parameter
[a]	Denotes an optional parameter
{a, b, c}	Denotes a selection of mandatory parameters (exactly one must be present)
[{a, b, c}]	Selection of optional parameters (either exactly one or none must be present)

Values

<integer>	An integer value
<number>	A floating point value
<string>	A string literal
<table>	A table
<var>	Variable type

Weiss Robotics GmbH & Co. KG
In der Gerste 2
D-71636 Ludwigsburg, Germany
e-mail: office@weiss-robotics.com

For further information and other products from Weiss Robotics, please visit our homepage at **http://www.weiss-robotics.com**.