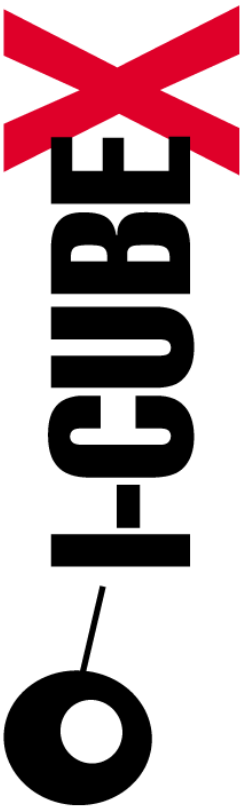

USB-microDig firmware v7.2

I-CubeX Reference

I-CubeX: The ultimate MIDI controller!

Infusion Systems Ltd.
2033 Vendome avenue
Montreal, QC
Canada H4A 3M4
Tel: (514) 484-5850
Fax: (514) 484-5852
Email: info@infusionsystems.com
http://www.infusionsystems.com

2 November 2015

I-CubeX USB-microDig firmware v7.2

INTRODUCTION

The USB-microDig is a thumb-sized hardware device that encodes analog voltage signals, such as generated by sensors, to music industry compliant MIDI messages with high resolution.

The USB-microDig also includes 8 digital outputs that can be used to control actuators using the top (4th) row on the sensor input connector. Use any of our many free software tools, such as the iCube and oCube Max/MSP plugins, to interface the USB-microDig to your software application.

The firmware of the USB-microDig enables it to operate in both stand-alone mode (sensor data is processed before it is transmitted) as well as host mode (raw sensor data is transmitted). Stand-alone mode includes various sensor processing and mapping features such as gesture recognition that can be conveniently configured using our free editor.

To communicate with the USB-microDig you need to connect its USB cable to your computer and install the supplied driver available for download from our web site (Windows or MacOS). You can then establish a connection with the USB-microDig by using our free Connect or BlueMIDI application or use the virtual COM port (VCP) with your own application (such as the serial object in Max/MSP). If you are using the VCP the COM port setting of the USB-microDig is 115200 bps, no parity, 1 stop bit, 8 data bits, hardware data flow control enabled.

The USB-microDig is self powered and provides 5V power to the connected sensors/actuators. It is registered as a device that uses 300mA, so if the maximum allowed current has been exceeded for the USB hub where the USB-microDig is connected you might get a system message on your computer warning you about this or the hub will simply shut the power off. In this case you will need to get a powered hub so that it can provide sufficient current to the USB-microDig. The USB-microDig is compliant with the USB v2.0 specification and uses the full-speed (12Mbps) transfer rate.

In this documentation the communication protocol used by the USB-microDig is based on the MIDI protocol. Despite the fact that the USB-microDig has a different baud rate, uses a serial port and has no MIDI cables, you can use it as a MIDI device by running our free serial port to MIDI bridging software that makes the serial port data available at a MIDI port on the computer. This way you can use any MIDI application to communicate with the USB-microDig. The “MIDI input” and “MIDI output” expressions used in this document therefore refer to a communication setup as established between the USB-microDig and the computer running our serial-MIDI bridging software. You can turn it into a stand-alone MIDI (output-only) device by using our USB-microMIDICable.

This document describes the I-CubeX USB-microDig communication protocol v7.2. The MIDI implementation version number is the same as the firmware version number and can be obtained by sending a DUMP VERSION command to the USB-microDig. If your USB-microDig’s firmware is older than v7.2 you can most likely upgrade it to v7.2 using our software, freely available from our website.

This specification is divided into general, stand-alone mode and dual mode sections addressing programming commands as understood by the USB-microDig and transmitted messages as sent by the USB-microDig.

Mode of operation

The USB-microDig can function in host mode, or stand-alone mode. Host mode is intended for using the USB-microDig with a computer that processes the sensor data and/or controls the actuator outputs, while stand-alone mode is intended for using the USB-microDig either on its own, where sensor data is mapped to actuator outputs, or together with other MIDI capable devices.

Stand-alone mode allows the USB-microDig to be used with its internal signal processing so that another MIDI capable device (which could be a computer) can directly use the calculated end result. Processing and mapping of sensor data, and control of actuator outputs happens in the USB-microDig itself, so once it has been programmed using our configuration software (free to download from our website) it can run by itself or it can be patched to MIDI-capable software or devices. This mode allows sensor inputs to be mapped to channel voice MIDI messages. Stand-alone mode commands (including STREAM and INTERVAL), contrary to host mode commands, change the settings stored in non-volatile memory as well as volatile memory. Stand-alone mode commands are executed in both modes of operation, even though in host mode no channel voice MIDI messages are output to reflect any changed settings.

Host mode requires that the USB-microDig is connected with a host computer to control and store the USB-microDig settings, and process the sensor data and/or control the actuator outputs. In host mode unprocessed, raw sensor values are output only as system exclusive MIDI messages (ie. no stand-alone mode type channel voice MIDI messages processed from sensor values are output). When using Max/MSP software, host mode allows multiple iCube / oCube Max/MSP objects for each USB-microDig (to this end, some of the commands are echoed back to the host computer). Host mode commands (except STREAM and INTERVAL), contrary to stand-alone mode commands, do not change the settings stored in non-volatile memory but only the settings stored in volatile memory. Generally, host mode commands have the same effect in either mode of operation – see for details below.

In host mode the USB-microDig can detect when the COM port is open by the use of hardware data flow control (using the RTS and CTS pins of the COM port, see also the COM command below). If the software application you are using doesn't have hardware data flow control (such as the Max/MSP serial object) then the COM port can be open but the USB-microDig pauses data processing and the IN and OUT LED on the USB-microDig will keep on flashing once a second until it receives a byte (that your software application sent). Once a byte has been received data processing resumes and the IN and OUT LED will only be ON momentarily once every second when data is transmitted as per their definition above, even after the COM port is closed.

If the software application uses hardware data flow control, and the USB-microDig is in host mode, when the USB-microDig COM port is closed (by default when the USB-microDig is being connected) the USB-microDig pauses data processing and the IN and OUT LED will blink again once every second.

When the USB-microDig is in stand-alone mode data processing is never paused, it ignores data flow control and the IN and OUT LED will not blink once a second but only when data is transmitted.

To set the USB-microDig mode, use the SET MODE command.

LED status indications

IN LED definition

The USB-microDig will give a visual feedback for any data received from it's USB interface. Each time data is received from the interface the IN LED will be ON momentarily.

OUT LED definition

The USB-microDig will give a visual feedback for any data sent to it's USB interface. Each time data is sent to the interface the OUT LED will be ON momentarily.

POWER LED definition

The USB-microDig will give visual feedback when power is received from the USB connector. When the power at the sensor input connector is 5V the POWER LED is normally lit. When the power at the sensor input connector is below 5V because of a short circuit or an excessive supply current, the POWER LED will be dimmed.

IN and OUT LED blinking

If the USB-microDig is in host mode, the IN and OUT LED on the USB-microDig will blink once a second if the COM port is closed or until it receives a byte.

All LEDs ON

If all LEDs are ON, the USB-microDig has encountered an unexpected error. Power it off and then on again to reset it.

Data Format

Infusion Systems I-CubeX USB-microDig MIDI implementation uses the following data format for all system exclusive messages:

Byte	Description
240 (F0h)	System Exclusive Status
125 (7Dh)	Manufacturer ID
{DEV}	Device ID
{CMD}	Command or Message ID
[BODY]	Main Data
247 (F7h)	End of System Exclusive

Manufacturer ID

The manufacturer identifies the manufacturer of a MIDI device. Infusion Systems uses the number 125 (7Dh). Infusion Systems does not own the manufacturer ID 125 (7Dh). This ID is usually reserved for general educational

and/or research equipment. Please ensure that you do not have any devices with manufacturer ID 125 (7Dh) in your MIDI chain.

Device ID

This ID identifies a specific USB-microDig in setups with multiple USB-microDigs. This setting should remain 0 for use of the USB-microDig in stand-alone mode. Host mode allows for device IDs other than 0.

Command and Message IDs and [Main Data]

The Command and Message IDs and [Main Data] form the protocol for communication with the USB-microDig. The protocol consists of commands that are sent to the USB-microDig and messages that are sent by the USB-microDig. The Command and Message ID description and [Main Data] formats are described starting on the next page.

General Command and Message IDs and [Main Data]

This section describes the messages and commands for resetting the USB-microDig, obtaining version information, setting the mode of operation (stand-alone or host mode), setting the device ID, setting MIDI out operation, setting running status and obtaining COM port status. These commands have the same effect both in stand-alone and host mode. This section also describes system status messages.

Programming commands

MIDI system reset command

The I-CubeX USB-microDig can be reset using the single byte MIDI system reset command (255, FFh) in exactly the same way as when using the RESET command (34,22h). A RESET ACK (35,23h) will be sent after completion of the reset command. The MIDI system reset command can be sent anytime during any message. Any other MIDI system real-time messages that are received in the USB-microDig MIDI input are ignored.

Command ID: MUTE (32, 20h)

The MUTE command stops the sampling of all sensor inputs. It can be used in both modes of operation (stand-alone, host). After completion of the MUTE command, the USB-microDig does not send out any MIDI messages representing sensor values. After a RESET, the USB-microDig is un-muted by default. The MUTE command is a toggle. Sending it the first time after power-up mutes all active sensor inputs, sending it a second time un-mutes them, and so forth. The USB-microDig does not send out any messages upon completion of this command.

There is no [BODY] associated with this command.
Example: 240, 125, 0 {DEV}, 32 {MUTE}, 247 (F0h, 7Dh, 00h, 20h, F7h)

Command ID: SET MUTE (50, 32h)

The SET MUTE command stops the sampling of all sensor inputs. It can be used in both modes of operation (stand-alone, host). After completion of the SET MUTE command, the USB-microDig does not send out any MIDI messages representing sensor values. After a RESET, the USB-microDig is un-muted by default. The USB-microDig does not send out any messages upon completion of this command.

There [BODY] of the MUTE command:

0xxxxxxx:	xxxxxxx = 0; all sensor inputs un-muted
	xxxxxxx = [1..127]; all sensor inputs muted

Example: 240, 125, 0 {DEV}, 50 {SET MUTE}, 1, 247 (F0h, 7Dh, 00h, 32h, 01h, F7h)

Command ID: RESET (34, 22h)

The RESET command resets the USB-microDig, i.e. all internal circuitry and system variables stored in volatile memory are re-initialized. Non-volatile memory remains unchanged. The RESET command is always executed following a power-up. The RESET command does not change any stand-alone mode settings stored in non-volatile memory but reloads them in volatile memory for operational use (when in stand-alone mode).

When the USB-microDig is in *host mode*, the RESET command sets internal parameters stored in volatile memory to initial values as specified below.

When the USB-microDig is in *stand-alone* mode, the RESET command retrieves the settings as stored in non-volatile memory and sets other internal parameters as specified below.

The RESET command can also be initiated by sending a single-byte MIDI system reset (255, FFh). The single-byte MIDI system reset message can be sent anytime during any MIDI message.

Upon completion of the RESET command the RESET ACK message is sent out by the USB-microDig *once* (host mode) or *twice* (stand-alone mode) depending on the mode of operation of the USB-microDig.

There is no [BODY] associated with this command.

Example (this is the same as sending this single byte: 255 (FFh)):

240, 125, 0 {DEV}, 34 {RESET}, 247 (F0h, 7Dh, 00h, 22h, F7h)

Initial values for RESET executed with USB-microDig set to host mode:

Mode of operation:	0; host mode
MIDI out activation status:	retrieved from non-volatile memory (see MIDI OUT command)
System exclusive ID:	retrieved from non-volatile memory (see SET ID command)
Sensor input muting:	0; off
Sensor activation status:	0; off
Sensor sampling resolution:	7 bit
Sensor sampling interval:	10 ms (see SET INTERVAL command)
Actuator activation status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator PWM status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator PWM interval:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)
Actuator PWM width:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)

Initial values for RESET executed with USB-microDig set to stand-alone mode:

Mode of operation:	1; stand-alone mode
MIDI out activation status:	retrieved from non-volatile memory (see MIDI OUT command)
System exclusive ID:	retrieved from non-volatile memory (see SET ID command)
Sensor input muting:	0; off

Other sensor input settings:	retrieved from non-volatile memory (see EDIT CONFIG command)
Actuator activation status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator PWM status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator pulse interval:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)
Actuator pulse width range:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)

Command ID: OUTPUT INIT (60, 3Ch)

The OUTPUT INIT command set the USB-microDig's actuator outputs to their power-up or reset status. Non-volatile memory remains unchanged. The OUTPUT INIT command does not change any stand-alone mode settings stored in non-volatile memory but reloads them in volatile memory for operational use. The OUTPUT INIT command is always executed following a power-up. Upon completion of the OUTPUT INIT command the same message is sent out by the USB-microDig.

There is no [BODY] associated with this command.

Example:

240, 125, 0 {DEV}, 60 {OUTPUT INIT}, 247 (F0h, 7Dh, 00h, 3Ch, F7h)

Initial values for INIT OUTPUTS:

Actuator activation status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator PWM status:	retrieved from non-volatile memory (see SET OUTPUT INIT command)
Actuator pulse interval:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)
Actuator pulse width range:	retrieved from non-volatile memory (see SET OUTPUT PULSE command)

Command ID: DUMP VERSION (71, 47h)

This command requests the USB-microDig to send out its serial number and firmware version number. Upon completion of the command the USB-microDig sends out the VERSION message.

There is no [BODY] for the DUMP VERSION command.

Command ID: SET MODE (90, 5Ah)

This command sets the USB-microDig operation mode. When switching the mode of operation, all settings stored in non-volatile memory remain unchanged but all settings stored in volatile memory are re-initialized to the values stored in non-volatile memory. When switching to host mode the initial values for host mode are used (see the RESET

command). When switching to stand-alone mode the initial values for stand-alone mode are used (see the RESET command). The default value for the mode of operation is stand-alone mode. Upon completion of the command the USB-microDig sends out the MODE message.

The [BODY] of the SET MODE command is as follows:

0000000x: x = 0; host mode
x = 1; stand-alone mode

Command ID: DUMP MODE (91, 5Bh)

This command requests the USB-microDig to send out its current mode of operation, which may be different from the mode stting stored in non-volatile memory. The USB-microDig can function in host mode (for use with the iCube plugin for Max/MSP), or stand-alone mode (otherwise). See the SET MODE command for more information on the mode of operation. Upon completion of the command the USB-microDig sends out the MODE message.

There is no [BODY] for the DUMP MODE command.

Command ID: SET ID (92, 5Ch)

The SET ID command sets the system exclusive device ID of the USB-microDig, enabling use of up to 128 USB-microDigs in one MIDI chain. The SET ID command can be sent with any device ID in the header of the system exclusive message, ie. the device ID will be set to the new value regardless of the current device ID of the USB-microDig receiving the SET ID command. The default value is 0. Upon completion of the command the USB-microDig sends out the same command.

The [BODY] of the SET ID command:

0xxxxxxx: xxxxxxx = [0..127]; new system exclusive device ID

Command ID: MIDI OUT (93, 5Dh)

This command sets the MIDI out status of the USB-microDig. The default value is on, ie. MIDI out is enabled. The MIDI OUT command configures the USB-microDig to set the baud rate of the USB-microDig COM port to 31.25 kbps upon power-up when no USB port is connected. Instead, if a USB-microMIDICable is attached to the USB-microDig it's MIDI messages can be received by another MIDI device. Upon completion of the command the USB-microDig sends out the same command.

The [BODY] of the MIDI OUT command is:

0xxxxxxx: xxxxxxx = 0; MIDI out off
xxxxxxx = [1..127]; MIDI out on

Command ID: DUMP MIDI OUT (96, 60h)

This command requests the USB-microDig to send out the MIDI out status. See the MIDI OUT command for more information on the MIDI out feature. Upon completion of the command the USB-microDig sends out the DUMP MIDI OUT message.

There is no [BODY] for the DUMP MIDI OUT command.

Command ID: MIDI RUNNING STATUS (94, 5Eh)

This command enables / disables MIDI running status.

When this is enabled the USB-microDig will always send the MIDI status byte each time it sends a MIDI message. For example, if the running status feature of the USB-microDig is ON then the output MIDI message from the USB-microDig for two MIDI note-on command (9xh where x is a number from 0 to 15 representing the MIDI channel number between 1 and 16) on MIDI channel 1 will be “90h n1 v1 90h n2 v2”. Where 90h is the running status byte for a note -on on channel 1, n1 & n2 are the note number values between 0 to 127 (for MIDI note-on 1 and 2) and v1 & v2 are the velocity values between 0 and 127 for MIDI note-on 1 and 2.

When the MIDI running status feature is disabled the USB-microDig will only send a running status byte when it is different from the last one that was sent. For example the running status for two notes would be: “90h n1 v1 n2 v2”.

This feature is useful when the data stream is large. It removes unwanted repetition of the current status so that this can slightly reduce the latency and the amount of MIDI data flow.

The [BODY] of the MIDI RUNNING STATUS command is:

0xxxxxxx:	xxxxxxx = 0; MIDI running status disabled
	xxxxxxx = [1..127]; MIDI running status enabled

Command ID: DUMP RUNNING (95, 5Fh)

This command requests the USB-microDig to send out whether running status is enabled or not. See the RUNNING command for more information. Upon completion of the command the USB-microDig sends out the DUMP RUNNING message.

There is no [BODY] for the DUMP RUNNING STATUS command.

Command ID: DUMP COM (111, 6Fh)

In host or stand-alone mode the host can send a request to the USB-microDig to get its COM port status.

There is no [BODY] for the COM command.

Example:

240, 125, 0 {DEV}, 111 {COM}, 247 (F0h, 7Dh, 00h, 6Fh, F7h)

Transmitted messages

Message ID: RESET ACK (35, 23h)

The RESET ACK message provides acknowledgement that the USB-microDig has been reset. A reset is performed when the USB-microDig is powered up and that COM port connection is established, or when a RESET command is sent to it. If the USB-microDig is set to host mode and a RESET command is sent, the USB-microDig will send out the RESET ACK message once (visible as one blink of the OUT LED). If in stand-alone mode the RESET ACK message will be sent out twice with a 1s interval in between the two messages (visible as two blinks of the OUT LED).

There is no [BODY] for the RESET ACK message.

Message ID: STATUS (37, 25h)

The STATUS message indicates an error in the system. The [BODY] of the STATUS message indicates the type of problem encountered.

The [BODY] of the reset message:

0xxxxxxx:	xxxxxxx = 90 (5Ah);	configuration setting error
	xxxxxxx = 92 (5Ch);	MIDI protocol error / Invalid Command (invalid command number, bad number of arguments or invalid arguments in command)

xxxxxxx = 94 (5Eh);	MDI byte has been scrambled
xxxxxxx = 95 (5Fh);	receive buffer is full, data may be lost (too much MIDI data is being sent to the USB-microDig too quickly)

Example:

240, 125, 0 {DEV}, 37 {STATUS}, 95 {DATA}, 247 (F0h, 7Dh, 00h, 25h, 5Fh, F7h) This message indicates that too much MIDI data is being sent to the USB-microDig too quickly.

Message ID: VERSION (71, 47h)

This message contains the USB-microDig firmware version number, hardware board version number as well as the serial number.

The [BODY] of the VERSION message is as follows:

0xxxxxxx:	xxxxxxx = [0..127]; firmware version number * 10
0aaaaaaa:	aaaaaaa = [0..99]; hardware board vers number * 10
0bbbbbbb:	bbbbbbb = [0..99]; hardware board vers number decimals * 1000
0ccccccc:	ccccccc = [0..99]; 1st serial number digits
0ddddddd:	ddddddd = [0..99]; last serial number digits

Example:

240, 125, 0 {DEV}, 71 {VERSION}, 72 {x}, 72 {a}, 0 {b}, 01 {c}, 23 {d}, 247 (F0h, 7Dh, 00h, 47h, 46h, 46h, 00h, 01h, 17h, F7h)

The firmware version for the USB-microDig is 72 / 10 = 7.2. The hardware board version number is 72 / 10 + 0 / 1000 = 7.20. The serial number is 0123. These numbers should correspond with numbers printed on the sticker at the bottom of the USB-microDig – if they don't correspond the USB-microDig may have been upgraded to an updated firmware version after its purchase.

Message ID: MODE (91, 5Bh)

This message contains the USB-microDig operation mode. The USB-microDig can function in host mode (for use with Max/MSP for example), or in stand-alone mode (with most MIDI applications). See the SET MODE command for more information on the mode of operation.

The [BODY] of the MODE message is as follows:

0000000x:	x = 0; host mode x = 1; stand-alone mode

Message ID: MIDI OUT (96, 60h)

This message contains the USB-microDig MIDI out status. See the DUMP MIDI OUT command for more information on the MIDI out feature.

The [BODY] of the MIDI OUT message is as follows:

0000000x: x = 0; MIDI out disabled
x = 1; MIDI out enabled

Message ID: RUNNING (95, 60h)

This message contains the activation status of the USB-microDig running status. See the DUMP RUNNING command for more information.

The [BODY] of the RUNNING message is as follows:

0000000x: x = 0; running status disabled
x = 1; running status enabled

Message ID: COM (111, 6Fh)

This message contains the COM port status of the USB-microDig. See the DUMP COM command for more information.

The [BODY] of the COM message is as follows:

00000xyz: x = 0; DTR high
x = 1; DTR low
y = 0; RTS low
y = 1; RTS high
z = 0; SUSP low (USB enabled)
z = 1; SUSP high (USB disabled)

Stand-alone mode Command and Message IDs and [Main Data]

This section describes the programming protocol, consisting of commands sent to the USB-microDig and messages sent by the USB-microDig, specifically for stand-alone mode.

Programming commands

Command ID: EDIT NAME (100, 64h)

The EDIT NAME command stores the name (as ASCII encoded characters) of the configuration currently stored in the USB-microDig. Upon completion of the command the USB-microDig sends out a NAME message.

The [BODY] of the EDIT NAME command is:

00000001:	1; configuration number
0aaaaaaa:	aaaaaaa = [0..127]; 1st character, default = “U” (55h)
0bbbbbb:	bbbbbb = [0..127]; 2nd character, default = “S” (53h)
0ccccccc:	ccccccc = [0..127]; 3rd character, default = “B” (42h)
0ddddddd:	ddddddd = [0..127]; 4th character, default = “-” (2Dh)
0eeeeeee:	eeeeeee = [0..127]; 5th character, default = “u” (75h)
0fffffff:	fffffff = [0..127]; 6th character, default = “D” (44h)
0gggggggg:	gggggggg = [0..127]; 7th character, default = “i” (69h)
0hhhhhhh:	hhhhhhh = [0..127]; 8th character, default = “g” (67h)

Command ID: DUMP NAME (101, 65h)

The DUMP NAME command is used to obtain the name of the current configuration of the USB-microDig. Upon completion of the command the USB-microDig sends out a NAME message.

The DUMP NAME command has the following [BODY]:

00000001: 1; configuration number

Command ID: CLEAR CONFIG (105, 69h)

The CLEAR CONFIG command sets all settings stored in non-volatile memory to default values and re-initializes all internal parameters to the initial values for a USB-microDig set to stand-alone mode (see RESET – note that the RESET is not executed). See the EDIT NAME, EDIT CONFIG commands for default values of settings stored in non-volatile memory. Upon completion of the command the USB-microDig sends the same command.

The [BODY] of the CLEAR CONFIG command:

00000001: 1; configuration number

Default values for each sensor input and all actuator outputs are re-instated:

MIDI out:	1; on
MIDI running status	1; on
System exclusive ID:	0
Mode of operation:	1; stand-alone
Configuration name:	“USB-uDig”
Sensor sampling rate:	10 ms
Sensor input MIDI mapping type:	3; control-change
Sensor input impulse end notification:	0; off
Sensor input impulse maximum set to constant:	0; off
Sensor input cont. and/or impulse signal differentiation:	0; off
Sensor input continuous signal averaging:	0; off
Sensor input impulse signal processing:	0; off
Sensor input continuous signal processing:	0; off
Sensor input threshold:	0
Sensor input ceiling:	127
Sensor input noise gate:	0; off
Sensor input time window:	0; off
Sensor input function number:	0; off
Sensor input function parameter 1:	0;
Sensor input function device address / parameter 2:	0;
Sensor input to Actuator output mapping:	0; off
Actuator output MIDI mapping type:	1; note-on
Actuator output MIDI channel mapping:	0
Actuator output MIDI base mapping:	64
Actuator output response mode:	1; toggle mode

Actuator output MIDI value threshold:	1
Actuator output status:	0; off
Actuator output PWM status:	0; off
Actuator output pulse interval:	20 ms
Actuator output pulse width range:	1-2 ms

Command ID: EDIT CONFIG (106, 6Ah)

Each of the sensor inputs of the USB-microDig can be edited using the EDIT CONFIG command. Upon completion of the command the USB-microDig is re-initialized with the new settings and the USB-microDig sends out a CONFIG message. The EDIT CONFIG command can be used to start sending out MIDI messages calculated from sensor values immediately while the USB-microDig is powered, but it is also possible to send a STREAM command to the USB-microDig. Activating either or both signal processing methods will enable the sending of MIDI messages. To change the sampling interval use the INTERVAL command. The RESET command does not change any of the settings as stored by the EDIT CONFIG command in non-volatile memory.

Editing sensor inputs

To edit a sensor input of the USB-microDig the [BODY] of the EDIT CONFIG command consists of the following bytes:

00000001:	1; configuration number
00000aaa:	aaa = [0..7]; sensor input number
0bbbcccc:	bbb = [0..6]; MIDI mapping type (0 = note-off, 1 = note-on, 2 = key-pressure, 3 = control-change, 4 = program-change, 5 = after-touch, 6 = pitch-bend) cccc = [0..15]; MIDI channel
0ddddddd:	ddddddd = [0..127]; note number (mapping type 0..2), controller number (mapping type 3), irrelevant for mapping type 4..6
00efghij:	e,f,g,h,i,j determine signal processing settings, see notes for combining e = [0,1]; impulse end notification (0 = off, 1 = on) f = [0,1]; impulse maximum/minimum constant (0 = off, 1 = on) g = [0,1]; continuous and/or impulse signal differentiation (0 = off, 1 = on) h = [0,1]; continuous and/or impulse signal smoothing (0 = off, 1 = on) i = [0,1]; impulse signal processing (0 = off, 1 = on) j = [0,1]; continuous signal processing (0 = off, 1 = on)
0kkkkkkk:	kkkkkkk = [0..127]; sensor input threshold
0mmmmmmm:	mmmmmmm = [0..127]; sensor input ceiling
0nnnnnnn:	nnnnnnn = [0..127]; noise gate
0pppqqqq:	ppp = [0..7]; represents constant value activated with {f} or time window if smoothing is activated with {h} qqqq = [0..15]; time window or smoothing factor

Default settings for each sensor input are:

Sensor input MIDI mapping type {bbb}:	3; control-change
Sensor input MIDI channel mapping {cccc}:	0
Sensor input MIDI ctl/note mapping {ddddddd}:	sensor input {aaaaa} + 1
Sensor input impulse end notification {e}:	0; off
Sensor input impulse maximum set to constant {f}:	0; off
Sensor input cont. and/or impulse signal diff. {g}:	0; off
Sensor input continuous signal smoothing {h}:	0; off
Sensor input impulse signal processing {i}:	0; off
Sensor input continuous signal processing {j}:	1; on
Sensor input threshold {kkkkkkk}:	0; minimum
Sensor input ceiling {mmmmmmm}:	127; maximum
Sensor input noise gate {nnnnnnn}:	1
Sensor input constant value {ppp}:	0
Sensor input time window or smoothing factor {qqqq}:	0; off

MIDI mapping settings {bbbcccc} in decimal with MIDI channel = 0:

0 = note-off
16 = note-on
32 = key-pressure
48 = control-change
64 = program-change
80 = after-touch
96 = pitch-bend

Signal processing operations ({efghij} in decimal):

1 = continuous signal output
2 = impulse (peak or dip) detection
3 = impulse detection, then continuous output
4 = no signal processing (output disabled)
5 = continuous smoothed output
6 = impulse detection, peak-peak timing output (as milliseconds in pitch-bend message, scaled otherwise)
7 = impulse detection, smoothed, then continuous smoothed output
8 = no signal processing (output disabled)
9 = continuous differentiated output (ie. the difference between the last two samples)
10 = impulse detection, differentiated
11 = impulse detection differentiated, then continuous differentiated output
12 = no signal processing (output disabled)
13 = continuous output, smoothed and then differentiated
14 = impulse detection, smoothed and then differentiated
15 = impulse detection, smoothed and differentiated, then continuous smoothed, differentiated output

16 = no signal processing (output disabled)
17 = continuous signal output ({f} is ignored)
18 = impulse detection, output as a constant value
19 = impulse detection, output as a constant value, then continuous output
20 = no signal processing (output disabled)
21 = smoothed continuous output ({f} is ignored)
22 = impulse detection, peak-peak timing output as BPM (beats per minute)
23 = impulse detection, output as a constant value, then continuous smoothed output
24 = no signal processing (output disabled)
25 = continuous differentiated output ({f} is ignored)
26 = impulse detection, smoothed, output as a constant value
27 = impulse detection, output as a constant value, then continuous differentiated output
28 = no signal processing (output disabled)
29 = continuous smoothed output, differentiated ({f} is ignored)
30 = impulse detection, output as a constant value ({gh} are ignored)
31 = impulse detection, output as a constant value, then continuous smoothed output, differentiated
32 = no signal processing (output disabled)
33 = continuous output ({e} is ignored)
34 = impulse detection with end of impulse notification
35 = impulse detection with end of impulse notification, then continuous output
36 = impulse detection, smoothed, with end of impulse notification
37 = continuous smoothed output ({e} is ignored)
38 = impulse detection, single peak-end timing output
39 = impulse detection, single peak-end timing output, then continuous output
40 = no signal processing (output disabled)
41 = continuous differentiated output ({e} is ignored)
42 = impulse detection, differentiated, with end of impulse notification
43 = impulse detection, with continuous differentiated output and end of impulse notification
44 = no signal processing (output disabled)
45 = continuous smoothed, differentiated output ({e} is ignored)
46 = impulse detection, single peak-end timing output ({g} is ignored)
47 = impulse detection, with continuous differentiated output, then single peak-end timing output
48 = no signal processing (output disabled)
49 = continuous output ({ef} are ignored)
50 = impulse detection, output as a constant value, with end of impulse notification
51 = impulse detection, output as a constant value, with end of impulse notification, then continuous output
52 = no signal processing (output disabled)
53 = continuous smoothed output ({ef} are ignored)
54 = impulse detection, single peak-end timing output, smoothed
55 = impulse detection, with continuous output, then single peak-end timing output, smoothed
56 = no signal processing (output disabled)
57 = continuous differentiated output ({ef} are ignored)
58 = impulse detection, single peak-end timing output ({f} is ignored)
59 = impulse detection, with continuous differentiated output, then single peak-end timing output ({f} is ignored)
60 = no signal processing (output disabled)
61 = continuous smoothed, differentiated output ({ef} are ignored)
62 = impulse detection, single peak-end timing output, smoothed ({f} ignored)
63 = impulse detection, with continuous differentiated output, then single peak-end timing output, smoothed ({f} are ignored)

Mapping sensor inputs

Impulse signal processing is useful when the sensor signal can be characterized as an impulse, ie. as either rising from below a minimum value to a maximum value after which it eventually drops below a certain minimum value, or as dropping below a minimum value after which it eventually rises above a certain maximum value. Subsequent impulses may appear asynchronously.

Continuous signal processing is useful when the sensor signal can be characterized as a continuously varying signal without any specific start or end.

The impulse signal processing settings are:

- The activation status {i} which will also start sampling of the sensor input.
- The time window setting {qqqq} within which {qqqq} + 1 sensor values are compared so as to find the highest value (peak search, where the threshold {kkkkkkk} is smaller than {mmmmmmm}) or lowest value (dip search, where the threshold {kkkkkkk} is greater than {mmmmmmm}).
- The end notification setting {e} which provides a way to determine the duration of the impulse.
- The maximum/minimum constant setting {f}. For this setting the time window {qqqq} is not used. It provides a way to output each impulse maximum/minimum (peak/dip) as a constant of value 16 x {ppp} + 15, ie. There are only 8 constant values available.

If the sensor value rises above {mmmmmmm} ({mmmmmmm} greater than {kkkkkkk}) or drops below {mmmmmmm} (ie. {mmmmmmm} smaller than {kkkkkkk}) before {qqqq} + 1 samples have been obtained, the peak/dip search is stopped and is immediately output as peak/dip value.

The continuous signal processing settings are:

- The activation status {j} which will also start sampling of the sensor input
- The smoothing setting {g} which enables calculation of the exponential moving average of the sensor value.
- The smoothing factor setting {qqqq} (the higher the value, the more smoothing is applied).
- The differentiation setting {g} which enables calculation of the difference between the current and last output value.

Continuous signal processing can be combined with impulse signal processing and will be useful for sensor signals that have an onset with a local maximum or minimum after which the sensor value continues for some time to stay above the minimum or below the maximum before it eventually drops below the minimum or rises above the maximum.

Impulse signal processing can also be applied to continuously varying signals such as AC (alternating current) signals and recurring impulses of which only the peak (or dip) value is relevant. Deactivate impulse end notification, ie. set {e} = 0, to avoid sending out MIDI value zero.

In both impulse and continuous signal processing the 10-bit sensor value is scaled between the threshold {kkkkkkk} and ceiling {mmmmmmm}, and the resulting value

has to increase beyond the noise gate {nnnnnnn} (mapping types 1-5) or beyond the noise gate {nnnnnnn} times 4 (mapping type 6) before being output. After passing the noise gate the value itself or the absolute difference between the current and last calculated value can be output by activating the differentiation setting {g}.

If either impulse or continuous signal processing is active, the calculated values are represented by the 2nd data field of a MIDI channel voice note-off (header 80h), note-on (header 90h), key-pressure (header A0h) message with note number {ddddddd}, or by the 2nd data field of a control-change (header B0h) message with control number {ddddddd}, or by the 1st data field of a program-change (header C0h) or after-touch (header D0h) message or by the two data fields of a pitch-bend (header E0h) message.

If both impulse and continuous signal processing are active the sensor value's start and end are characterized using impulse signal processing and represented as one of the channel voice MIDI messages listed above, while the signal between start and end is characterized using continuous signal processing and always represented as the key-pressure value of a MIDI channel voice key-pressure (header A0h) message. Note that if smoothing is activated, {qqqq} will also be used as the smoothing factor, ie. {qqqq} + 1 samples will be taken for peak/dip detection and subsequently {qqqq} will be used as smoothing factor.

For example, if impulse signal processing is activated ({i} = 1), the threshold {kkkkkkk} is smaller or equal than the ceiling {mmmmmmm} and note-on mapping has been selected, then, once the sensor value rises above the threshold {kkkkkkk}, a note-on message with velocity value greater than zero will be sent and no new note-on message with velocity greater than zero will be sent until the sensor value has dropped below {kkkkkkk}. If continuous signal processing is activated as well ({j} = 1), then, after sending a note-on message, the sensor value will be represented by the key-pressure data field of a MIDI channel voice key-pressure message, until the sensor drops below {mmmmmmm} upon which a note-on message with velocity zero will be sent out.

If either impulse or continuous signal processing is activated, MIDI messages calculated as defined by the signal processing method(s) will be sent out. If both are deactivated no MIDI output is generated. MIDI running status is implemented in the MIDI output mapping when using stand-alone mode.

Mapping sensor inputs to actuator outputs

In order for actuator outputs to respond to sensor input the EDIT CONFIG command is used with a special value for the 2nd databyte in the [BODY]:

00000001:	1; configuration number
0x000yyy:	x = 1; input-output mapping flag yyy = [0..7]; sensor input number
0mn00zzz:	m = 0; input-output mapping disabled m = 1; input-output mapping enabled n = 0; trigger mode (normal output if PWM enabled) n = 1; toggle mode (inverted output if PWM enabled) zzz = [0..7]; actuator output number
0vvvvvvv:	vvvvvvv = [1..127]; on-threshold (maximum input if PWM enabled)
0wwwwwww:	wwwwwww = [0..126]; off-threshold (minimum input if PWM enabled)

Trigger mode means a sensor input value greater than or equal to the on-threshold {vvvvvvv} turns actuator output on and a message with velocity or data field smaller than or equal to the off-threshold {wwwwwww} turns actuator output off.

Toggle mode means a sensor input value greater than or equal to the on-threshold {vvvvvvv} or a sensor input value smaller than (and not equal to) the off-threshold {wwwwwww} toggles actuator output on and off.

If the actuator output that the sensor input is mapped to is pulse-width modulation enabled, the sensor input value is clipped between the thresholds and scaled to the difference between the thresholds. Toggle mode inverts the scaling.

Mapping MIDI input to actuator outputs

In order to program the actuator outputs the EDIT CONFIG command is used with a special value for the 2nd databyte in the [BODY]:

00000001:	1; configuration number
01111111:	127; actuator outputs flag
00aabbbb:	aa = [0..3]; MIDI mapping type (0 = note-off, 1 = note-on, 2 = key-pressure, 3 = control-change) bbbb = [0..15]; MIDI channel
0ccccccc:	ccccccc = [0..120]; base note number or control number
0000defg:	d,e,f,g are response mode bits for actuator outputs 8 (d) .. 5 (g), d,e,f,g = 0; trigger mode (normal output if PWM enabled) d,e,f,g = 1; toggle mode (inverted output if PWM enabled)
0000hijk:	h,i,j,k are response mode bits for actuator outputs 4 (h) .. 1 (k), h,i,j,k = 0; trigger mode (normal output if PWM enabled) h,i,j,k = 1; toggle mode (inverted output if PWM enabled)
0000mnpq:	m,n,p,q are initialisation bits for actuator outputs 8 (m) .. 5 (q) m,n,p,q = 0; actuator output is off after power-up or reset m,n,p,q = 1; actuator output is on after power-up or reset
0000rstu:	r,s,t,u are initialisation bits for actuator outputs 4 (r) .. 1(u) r,s,t,u = 0; actuator output is off after power-up or reset r,s,t,u = 1; actuator output is on after power-up or reset
0vvvvvvv:	vvvvvvv = [0..127]; MIDI value threshold (maximum MIDI value if PWM enabled)

Initialisation bits apply to settings upon power-up, which can also be set with the SET OUTPUT INIT command, including the pulse width modulation interval and width range parameters that apply to all actuators.

Default settings for all actuator outputs are:

Actuator output MIDI mapping type {aa}:	1; note-on
Actuator output MIDI sensor-actuator mapping {z}:	0; off
Actuator output MIDI channel mapping {bbbb}:	0
Actuator output MIDI base mapping {ccccccc}:	64
Actuator output response mode {d,e,f,g,h,i,j,k}:	0; trigger mode / normal output
Actuator output initialisation {m,n,p,q,r,s,t,u}:	1; on
Actuator output MIDI value threshold / max {vvvvvvv}:	1

Trigger mode means a note-off (header 80h), note-on (header 90h), key-pressure (header A0h) or control-change (header B0h) message with velocity or data field greater

than or equal to the MIDI value threshold {vvvvvvv} turns actuator output on and a message with velocity or data field equal to zero turns actuator output off.

Toggle mode means a note-off (header 80h), note-on (header 90h), key-pressure (header A0h) or control-change (header B0h) message with velocity or data field greater than or equal to the MIDI value threshold {vvvvvvv} toggles actuator output on and off.

If the actuator output that the sensor input is mapped to is pulse-width modulation enabled, the sensor input value is clipped and scaled by the MIDI value threshold. Toggle mode inverts the scaling.

Base note is the note number or control number which triggers or toggles actuator output 1.

The MIDI running status feature is available in the MIDI input mapping when using stand-alone mode.

Sensor-actuator mapping uses the configuration of the sensor input and the configuration of the actuator output. Regardless whether sensor-actuator mapping is enabled, the actuator output will still respond to the MIDI input mapping as per its configuration and the sensor input will still output MIDI messages as per its configuration.

Command ID: DUMP CONFIG (107, 6Bh)

The DUMP CONFIG command is used to obtain the current settings of the USB-microDig. Upon completion of the command the USB-microDig sends out a CONFIG message.

The USB-microDig will send out the configuration of a USB-microDig sensor input when using the DUMP CONFIG command with the following [BODY]:

00000001: 1; configuration number
00000aaa: aaa = [0..7]; sensor input number

The Digitizer will send out the configuration of each USB-microDig binary output as well as the sample interval value when using the DUMP CONFIG command with a special value for the 2nd data byte in the [BODY]:

00000001: 1; configuration number
01111111: 127; actuator outputs flag

Transmitted messages

Message ID: NAME (101, 65h)

The NAME message contains the name (as ASCII encoded characters) of the configuration currently stored in the USB-microDig.

The [BODY] of the NAME message is identical to the [BODY] of the EDIT NAME command.

Message ID: CONFIG (106, 6Ah)

The CONFIG message contains the configuration of a sensor input.

When receiving the configuration of a sensor input, the [BODY] of the CONFIG message is identical to the [BODY] of the EDIT CONFIG command.

Dual mode Command and Message IDs and [Main Data]

This section describes the programming protocol, consisting of commands sent to the USB-microDig to configure sensor inputs and actuator outputs and messages sent by the USB-microDig, for either host mode or stand-alone mode.

Command ID: STREAM (1, 01h)

The STREAM command sets the sensor input activation status, ie. whether USB-microDig is sampling the sensor input.

In stand-alone mode, only if a signal processing method has been selected (impulse and/or continuous signal processing, see EDIT CONFIG command) each sensor input of the USB-microDig can be turned on or off with the STREAM command. If no signal processing method has been selected the STREAM command has no effect. The activation status of the sensor input is stored in non-volatile memory as well as volatile memory for immediate use. Following the activation of a sensor input the MIDI message to which the sensor value is mapped as set using the EDIT CONFIG command, is sent out by the USB-microDig. After a RESET command in stand-alone mode the activation status is retrieved from non-volatile memory - each sensor input is turned on or off depending on whether any of the signal processing methods is activated.

In host mode the STREAM command has immediate effect and if the sensor input is activated, STREAM DATA messages are output. The activation status of the sensor input is only stored in volatile memory for immediate use. After a RESET command in host mode each sensor input is turned off.

Upon completion of the command the USB-microDig sends out the same message.

The [BODY] of the STREAM command consists of a single 7-bit byte with the following format :

0x000yyy: x = 1; on
x = 0; off
yyy = [0..7]; sensor input number, where the first sensor input number = 0, and the last (8^{th}) sensor input number = 7

Example:

In order to turn on the 3^{rd} sensor input, the following message is sent:

240, 125, 0 {DEV}, 1 {STREAM}, 66 {x = 1, yyy = 02}, 247 (F0h, 7Dh, 00h, 01h, 42h, F7h)

In order to turn off the same sensor input, the following message is sent:

240, 125, 0 {DEV}, 1 {STREAM}, 2 {x = 0, yyy = 02}, 247 (F0h, 7Dh, 00h, 01h, 02h, F7h)

Command ID: RES (2, 02h)

In host mode, each sensor input can either be sampled with 10-bit (hi-res) or 7-bit (lo - res) resolution. After a RESET, the resolution for each input is set to lo-res (in host mode). Upon completion of the command the USB-microDig sends out the same command. In stand-alone mode this command is only useful when sending the SAMPLE

command because in stand-alone mode sensor inputs are always sampled with 10-bit resolution.

The RES command's [BODY] is formatted the same as the STREAM command (the command ID is the only difference):

0x000yyy: x = 1; hi-res (10-bit mode)
x = 0; lo-res (7-bit mode)
yyy = [0..7]; sensor input number, where the first sensor input number = 0, and the last (8^{th}) sensor input number = 7

Example:

In order to turn the 2nd sensor input to hi-res mode, the following message is sent:

240, 125, 0 {DEV}, 2 {RES}, 65 {x = 1, yyy = 1}, 247 (F0h, 7Dh, 00h, 02h, 41h, F7h)

In order to turn the same sensor input to lo-res mode, the following message is sent:

240, 125, 0 {DEV}, 2 {RES}, 1 {x = 0, yyy = 1}, 247 (F0h, 7Dh, 00h, 02h, 01h, F7h)

Command ID: SAMPLE (4, 04h)

The SAMPLE command provides a snapshot of the data coming out of a single sensor input. Upon completion of the command the USB-microDig sends out the SAMPLE DATA message, whether in host or stand-alone mode.

The [BODY] of the SAMPLE command is composed of one byte - the sensor input number:

00000yyy: yyy = [0..7]; sensor input number, where the first sensor input number = 0, and the last (8^{th}) sensor input number = 7

Example:

In order to sample sensor input 5 (provided it is turned off), the following message is sent:

240, 125, 0 {DEV}, 4 {SAMPLE}, 4 {yyy}, 247 (F0h, 7Dh, 00h, 04h, 04h, F7h)

Command ID: FUNCTION (8, 08h)

The FUNCTION command enables a sensor-specific processing algorithm. Functions are available for both analog sensors as well as digital (I2C) sensors, where the latter also operate in stand-alone mode. The full list of functions can be found in section on sensor functions. When enabling host mode function numbers are reset (ie. set to 0) for each input and the other function settings are retrieved from non-volatile memory. When enabling stand-alone mode all function settings are retrieved from non-volatile memory. The command can be used in both host and stand-alone modes of operation. In both modes the function number is stored in volatile memory and in host mode the other function settings are stored in non-volatile memory while in stand-alone mode all function settings are stored in non-volatile memory. Upon completion of the FUNCTION command the USB-microDig sends out the same message.

In host mode raw sensor data will be output immediately in I2C DATA messages. In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. The sensor parameter's raw values as output in an I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0..127 (eg. control-change message) or 0..16383 (pitch-bend message).

The [BODY] of the FUNCTION command consists of four 7-bit bytes with the following format:

Analog sensors:

00000yyy:	yyy = [0..7]; sensor input number, where the first input number = 0 and the last (8th) input number = 7
0ddddddd:	ddddddd = [0-127]; function parameter 2 (FP2)
0nnnnnnn:	nnnnnnn = [1-63]; function number
0ppppppp:	ppppppp = [0-127]; function method or parameter 1 (FP1)

Digital (I2C) sensors:

00000yyy:	yyy = [0..6]; I2C port number, where the first port number = 0 and the last (7th) port number = 6
0ddddddd:	ddddddd = [0-126]; I2C device address
0nnnnnnn:	nnnnnnn = [64-127]; function number
0ppppppp:	ppppppp = [0-127]; function method or parameter (FP)

Examples:

In order to set I2C port 2 (input 3 and 4) to use the Orient3D function, method 1 (heading/pitch/roll) for an Orient3D sensor with address 96, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 2 {port number}, 96 {address of device}, 125 {Orient3D}, 1 {heading/pitch/roll}, 247 (F0h, 7Dh, 00h, 08h, 60h, 7Dh, 01h, F7h)

In order to turn the function off, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {address}, 0 {function}, 0 {parameter}, 247 (F0h, 7Dh, 00h, 08h, 00h, 00h, 00h, F7h)

Command ID: INTERVAL (3, 03h)

The USB-microDig's sampling interval is the time that the USB-microDig must wait until it samples again the enabled inputs. For example if the sampling interval is set to 100ms the USB-microDig will sample all of its enabled inputs and can be set to speed up or slow down data acquisition.

The sample interval applies to all sensor inputs, ie. it is not possible to set a different sample interval for each sensor input. The interval is set in milliseconds with 1ms being the lowest (ie. highest sampling rate), and 16383ms (about 16 seconds) being the highest (ie. lowest sampling rate).

When the INTERVAL command is sent in host mode the sampling interval is stored in volatile memory, while in stand-alone mode it is stored in both volatile as well as non-volatile memory. After a RESET in host mode, the sampling interval is set to its default value (see the RESET command), while in stand-alone mode it is retrieved from non-volatile memory.

The INTERVAL command [BODY] is composed of 2 bytes each containing a 7-bit number. The first byte is the most significant byte, while the second byte is the least significant byte. Upon completion of the command the USB-microDig sends out the same command.

The [BODY] of the INTERVAL command:

0xxxxxxx: xxxxxxx = [0..127]; sample interval MSB
0yyyyyyy: yyyyyyy = [0..127]; sample interval LSB

where sample interval = xxxxxxx * 128 + yyyyyyy

Example:

In order to set the sampling interval to 1 second (1000 ms), the following command is sent:

240, 125, 0 {DEV}, 3 {INTERVAL}, 7 {xxxxxxx}, 104 {yyyyyyy}, 247 (F0h, 7Dh, 00h, 03h, 07h, 68h, F7h)

The sample interval is 7 * 128 + 104 = 1000

Command ID: SEND INTERVAL (13, 0Dh)

The SEND INTERVAL command can be used to get the sampling interval. The command can be used in both host and stand-alone modes of operation. Upon completion of the SEND INTERVAL command the USB-microDig sends out the same message as the INTERVAL command.

The SEND INTERVAL command has no [BODY].

Example:

240, 125, 0 {DEV}, 13 {SEND INTERVAL}, 247 (F0h, 7Dh, 00h, 0Dh, F7h)

Command ID: SEND FUNCTION (18, 12h)

The SEND FUNCTION command can be used to get the function settings data for a sensor input or I2C port. The command can be used in both host and stand-alone modes of operation. In host mode it retrieves the function number from volatile memory and the other function settings from non-volatile memory. In stand-alone mode it retrieves all function settings from non-volatile memory. Upon completion of the SEND FUNCTION command the USB-microDig sends out the same message as the FUNCTION command.

The [BODY] of the SEND FUNCTION command contains a single 7-bit byte:

Analog sensors:

00000yyy: yyy = [0..7]; sensor input number, where the first input number = 0 and the last (8^{th}) input number = 7

Digital (I2C) sensors:

00000yyy: yyy = [0..6]; I2C port number, where the first port number = 0 and the last (7^{th}) port number = 6

Example:

240, 125, 0 {DEV}, 18 {SEND FUNCTION}, 2 {input or port}, 247 (F0h, 7Dh, 00h, 18h, 02h, F7h)

Command ID: TABLE ENTRY (20, 14h)

The TABLE ENTRY command stores a lookup table entry in non-volatile memory. The command can be used in both host and stand-alone modes of operation. Upon completion of the TABLE ENTRY command the USB-microDig sends out the same message.

The [BODY] of the TABLE ENTRY command:

000nnnnn: nnnnn = [0..31]; table number
00000ddd: ddd = [0..7]; table entry index MSB
0ddddddd: ddddddd = [0..127]; table entry index LSB
00000eee: eee = [0..7]; table entry value MSB
0eeeeeee: eeeeeee = [0..127]; table entry value LSB

Example:

To store value 312 as entry 256 in table 8, ie. to convert sensor value 256 into 312, send the following message:

240, 125, 0 {DEV}, 21 {TABLE ENTRY}, 8 {table number}, 2 {entry index MSB}, 0 {entry index LSB}, 2 {entry value MSB}, 56 {entry value LSB}, 247 (F0h, 7Dh, 00h, 15h, 08h, 02h, 00h, 02h, 38h, F7h)

Command ID: SEND TABLE ENTRY (21, 15h)

The SEND TABLE ENTRY command retrieves a lookup table entry stored in non-volatile memory. The command can be used in both host and stand-alone modes of operation. Upon completion of the SEND TABLE ENTRY command the USB-microDig sends out the same message as the TABLE ENTRY command.

The [BODY] of the SEND TABLE ENTRY command:

000nnnnn: nnnnn = [0..31]; table number
00000ddd: ddd = [0..7]; table entry index MSB
0ddddddd: ddddddd = [0..127]; table entry index LSB

Command ID: SEND TABLE (31, 1Fh)

The SEND TABLE command retrieves a lookup table with 1024 values that's stored in non-volatile memory. The command can be used in both host and stand-alone modes of operation. Upon completion of the SEND TABLE command the USB-microDig sends out the same message as the TABLE ENTRY command, for each of the 1024 entries.

The [BODY] of the SEND TABLE command contains a single 7-bit byte:

000nnnnn: nnnnn = [0..31]; table number

Example:

To download table 4 from a USB-microDig, send the following message:

240, 125, 0 {DEV}, 31 {SEND TABLE}, 4 {table number}, 247 (F0h, 7Dh, 00h, 1Fh, 04h, F7h)

Command ID: SET OUTPUT PULSE (47, 2Fh)

The SET OUTPUT PULSE command sets the output pulse type as specified by its interval and width range and applies to all outputs at the same time. The default pulse is 1 – 2 ms long and repeats every 20 ms. It can be used to control a standard RC servo. Power for the servo has to be supplied by a separate power supply. The SET OUTPUT PULSE command can also configure the pulse to have no minimum pulse width. The command can be used in both modes of operation. After a RESET of the USB-microDig the common pulse interval and width as set by the SET OUTPUT PULSE command are retrieved from non-volatile memory. Upon completion of the command the digitizer sends out the same command with the actual set values (ie. if a desired setting is not permitted, the corrected setting will be used and sent back). To get the values stored in non-volatile memory, use the SEND OUTPUT PULSE command.

The [BODY] of the SET OUTPUT PULSE command consists of two 7-bit bytes with the following format:

0uuuuuuu: uuuuuuu = [0..127]; pulse interval MSB
0vvvvvvv: vvvvvvv = [5..127]; pulse interval LSB
0zzzzzzz: zzzzzzz = 0; pulse width range of 1-2 ms (RC servo mode)
zzzzzzz = [3..127]; pulse width range in ms

The default setting is the standard pulse for RC servos: a pulse interval of 20 ms and a pulse width range of 1-2 ms.

Example:

In order to set the output pulse interval to 15 ms and the width range to 0-10 ms to control the light level emitted by an LED connected to the actuator output, the following message is sent:

240, 125, 0 {DEV}, 47 {SET OUTPUT PULSE}, 15 {interval}, 5 {width range}, 247 (F0h, 7Dh, 00h, 0Fh, 05h, F7h)

In order to set the output pulse interval to 20 ms and the width range to 1-2 ms to control an RC servo, the following message is sent:

240, 125, 0 {DEV}, 47 {SET OUTPUT PULSE}, 20 {interval}, 0 {width range}, 247 (F0h, 7Dh, 00h, 2Fh, 2h, 1h, F7h)

Command ID: SET OUTPUT (48, 30h)

The SET OUTPUT command can be used to turn any of the 8 available actuator outputs on or off. If pulse is enabled/disabled with the SET OUTPUT command the pulse is also enabled/disabled at power-up but this can be changed by using the SET OUTPUT INIT command. If the pulse is enabled and no pulse width value is set to use at power-up, the pulse width to use at power-up is set to zero. Enabling the pulse overrides the output on/off setting. The command can be used in both modes of operation. After a RESET of the USB-microDig, all outputs' pulse statuses are retrieved from non-volatile memory as set by the SET OUTPUT INIT command. Upon completion of the command the digitizer sends out the same command. To get the values stored in non-volatile memory, use the SEND OUTPUT command.

The [BODY] of the SET OUTPUT command consists of two 7-bit bytes with the following format:

0xp00yyy:	x = 1; output on x = 0; output off p = 1; enable pulse (also at power-up) p = 0; disable pulse (also at power-up) yyy = [0..7]; actuator output number, where the first actuator output number = 0, and the last (8th) actuator output number = 7
0zzzzzzz:	zzzzzzz = [0..127]; pulse width 1-2 ms (ie. steps of approx. 8 microseconds)

Example:

In order to set the 2nd actuator output pulse width to 2.0ms, the following message is sent:

240, 125, 0 {DEV}, 48 {SET OUTPUT}, 97 {x = 1, p = 1, yyy = 1}, 127 {pulse width}, 247 (F0h, 7Dh, 00h, 30h, 61h, 7Fh, F7h)

The pulse width is 1 + 127*0.008 = 2.0ms

In order to turn the same actuator output on but disable the pulse, the following message is sent:

240, 125, 0 {DEV}, 48 {SET OUTPUT}, 65 {x = 1, p = 0, yyy = 1}, 0 {pulse width}, 247 (F0h, 7Dh, 00h, 30h, 41h, 00h F7h)

When disabling the pulse the pulse width value is ignored.

Command ID: SET OUTPUT INIT (49, 31h)

The SET OUTPUT INIT command sets the output status at power-up of the USB-microDig. If pulse is enabled/disabled with the SET OUTPUT command the pulse is also enabled/disabled at power-up but this can be changed by using the SET OUTPUT INIT command. If the pulse is enabled and no pulse width value is set to use at power-up, the

pulse width to use at power-up is set to zero. Enabling the pulse overrides the output on/off setting. The command can be used in both host and stand-alone modes of operation. All output settings are stored in both volatile and non-volatile memory. After a RESET of the USB-microDig, all outputs' pulse statuses are retrieved from non-volatile memory whether in stand-alone mode or in host mode. Upon completion of the command the digitizer sends out the same command. To get the values stored in non-volatile memory, use the SEND OUTPUT INIT command.

The [BODY] of the SET OUTPUT INIT command consists of two 7-bit bytes with the following format:

0xp00yyy:	x = 1; output on x = 0; output off p = 1; enable pulse p = 0; disable pulse yyy = [0..7]; actuator output number, where the first actuator output number = 0, and the last (8th) actuator output number = 7
0zzzzzzz:	zzzzzzz = [0..127]; pulse width 1-2 ms (ie. steps of approx. 8 microseconds)

Example:

In order to set the 2nd actuator output pulse width at power-up to 1.5ms, the following message is sent:

240, 125, 0 {DEV}, 49 {SET OUTPUT INIT}, 97 {x = 1, p = 1, yyy = 1}, 64 {pulse width}, 247 (F0h, 7Dh, 00h, 31h, 61h, 40h, F7h)

The pulse width is 1 + 64*0.008 = 1.5ms

In order to turn the same actuator output off and disable the pulse, the following message is sent:

240, 125, 0 {DEV}, 49 {SET OUTPUT INIT}, 1 {x = 0, p = 0, yyy = 1}, 0 {pulse width}, 247 (F0h, 7Dh, 00h, 31h, 01h, 00h, F7h)

When disabling the pulse the pulse width value is ignored.

Command ID: SEND OUTPUT PULSE (57, 39h)

The SEND OUTPUT PULSE command can be used to get the output pulse type as specified by its interval and width range and which applies to all outputs at the same time. The command can be used in both host and stand-alone modes of operation. Upon completion of the command the digitizer sends out the same message as the SET OUTPUT PULSE command but with the command ID byte set to 57 (39h).

The SEND OUTPUT PULSE command has no [BODY].

Example:

240, 125, 0 {DEV}, 57 {SEND OUTPUT PULSE}, 247 (F0h, 7Dh, 00h, 39h, F7h)

Command ID: SEND OUTPUT (58, 3Ah)

The SEND OUTPUT command can be used to get the actuator output status. The command can be used in both host and stand-alone modes of operation. Upon completion of the command the digitizer sends out the same message as the SET OUTPUT command but with the command ID byte set to 58 (3Ah).

The [BODY] of the SEND OUTPUT command consists of a single 7-bit byte with the following format:

0000xyyy:	x = 1; apply the command to all actuator outputs (yyy is ignored), ie. send 8 messages with the status for each actuator output yyy = [0..7]; actuator output number, where the first actuator output number = 0, and the last (8th) actuator output number = 7

Example:

240, 125, 0 {DEV}, 58 {SEND OUTPUT}, 1 { yyy = 1}, 247 (F0h, 7Dh, 00h, 3Ah, 01, F7h)

Command ID: SEND OUTPUT INIT (59, 3Bh)

The SEND OUTPUT INIT command can be used to get the actuator output status. The command can be used in both host and stand-alone modes of operation. Upon completion of the command the digitizer sends out the same message as the SET OUTPUT INIT command but with the command ID byte set to 59 (3Bh).

The [BODY] of the SEND OUTPUT INIT command consists of a single 7-bit byte with the following format:

0000xyyy:	x = 1; apply the command to all actuator outputs (yyy is ignored), ie. send 8 messages with the status for each actuator output yyy = [0..7]; actuator output number, where the first actuator output number = 0, and the last (8th) actuator output number = 7

Example:

240, 125, 0 {DEV}, 59 {SEND OUTPUT INIT}, 1 {yyy = 1}, 247 (F0h, 7Dh, 00h, 3Bh, 01h, F7h)

Command ID: I2C PORT (125, 7Dh)

The USB-microDig can be used to communicate with I2C devices in both host and stand-alone modes of operation. Basically, I2C is a 2 wire communication bus allowing to transfer digital information between a master (herein the USB-microDig) and a slave (a sensor for example). General information about the I2C bus can easily be found on internet and is not intended to be described here. The implementation of the I2C bus on the USB-microDig uses an input pair of 2 adjacent inputs for its DATA (DA) and CLOCK (CK) lines.

Using the "I2C PORT " command you can check I2C port 0 to 6 and set it to input pair 1-2, 2-3, 3-4, 4-5, 5-6, 6 -7 or 7-8 respectively. It's not possible to use input pair 8-1 or

1-3 for example since they are not adjacent. The port number is set according to the distance from input pair 1-2 (pair 1-2 being port 0). For example, to set the port to input pair 7-8 you need to send the I2C PORT command with port number 6 (6^{th} pair from input pair 1-2). The USB-microDig can have up to four I2C ports connected (side to side port 0, 2, 4 and 6).

The lowest number in an input pair (ex: 3 in pair 3-4) is connected to the data (DA) line and the highest input (ex: 4 in pair 3-4) is connected to the clock (CK) line. Both lines need to have a pull-up resistor connected to 5V (usually between 4.7K-10K Ohm) before the I2C PORT command can be sent to the USB-microDig. If no pull-up resistors are present (voltage not above 4V on DA and CK lines) the USB-microDig will respond to the I2C PORT command with a I2C PORT "port not opened" message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port). If the port has pull-up resistors then the USB-microDig will reply with the same message as the command. If no pull-up resistors are present on the I2C device, the USB-microDig's internal pull-up resistors of 20K-50K Ohm can be enabled by setting the 6^{th} bit in the port number byte of the "I2C PORT" command. Note that the value of the pull-up resistors in the USB-microDig may not be low enough for the I2C device to be recognized by the USB-microdig.

A port can have as many devices as there are different I2C addresses. Since a device only responds when its address is called the port could connect connect up to 128 I2C devices (address 00h to 7Fh). For the 4 available ports this would give a maximum of 128 x 4 = 512 devices ! The limit would however be the current consumption taken from the inputs (60mA max per port) of the USB-microDig.

The USB-microDig can only be used as an I2C master (ie. there is no I2C slave mode). The USB-microDig can both use analog sensors and communicate with I2C digital devices.The I2C clock speed is approximately 50kHz and cannot be changed. The I2C addressing is 7 bits only (no 10 bits address mode).

The I2C PORT command [BODY] is:

0xp00yyy:	x = 1; enable port x = 0; disable all I2C ports yyy = [0..6]; I2C port number, where port number = 0 corresponds with inputs 1-2 p = 1; enable internal pull-up resistors p = 0; disable internal pull-up resistors

Examples:

This command attempts to open port 3 (inputs 4-5) for I2C communications:

240, 125, 0 {DEV}, 125 {I2C PORT}, 67 { x = 1, yyy = 3, p = 0}, 247 (F0h, 7Dh, 00h, 7Fh, 43h, F7h)

This command enables the internal pull-up resistors for port 3 (inputs 4-5) and then attempts to open port 3 for I2C communications:

240, 125, 0 {DEV}, 125 {I2C PORT}, 99 {x = 1, yyy = 3, p = 1}, 247 (F0h, 7Dh, 00h, 7Fh, 63h, F7h)

Command ID: I2C WRITE (126, 7Eh)

The I2C WRITE command can be used to transfer data to a I2C device. Unless the I2C device has no pull-up resistors it is not necessary to use the I2C PORT command before sending the I2C WRITE command. The command can be used in both host and stand-alone modes of operation.

The I2C WRITE command [BODY] is:

00000xxx:	xxx = [0..6]; I2C port number
0aaaaaaa:	aaaaaa = [0..127]; I2C address of device
0000bbbb:	
0000cccc:	bbbbcccc = [0..255]; register number of device
0000dddd:	
0000eeee:	ddddeeee = [0..255]; data to write to register bbbbcccc

Example:

240, 125, 0 {DEV}, 126 {I2C WRITE}, 04 {port number}, 56 {address of device}, 0, 3 {register 3 = 0*16+3}, 1, 2 {data byte 18 = 1*16+2}, 247 (F0h, 7Dh, 00h, 7Eh, 38h, 00h, 03h, 01h, 02h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C WRITE message identical to the I2C WRITE command sent. If there was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (ie. in the above case the response would be F0h, 00h, 7Bh, 04h, 38h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with an I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Command ID: I2C WRITE SINGLE (124, 7Ch)

The I2C WRITE SINGLE command is similar to the I2C WRITE command but is used for I2C devices that do not use a register access byte. Unless the I2C device has no pull-up resistors it is not necessary to use the I2C PORT command before sending the I2C WRITE SINGLE command. The command can be used in both host and stand-alone modes of operation.

The I2C WRITE SINGLE command [BODY] is:

00000xxx:	xxx = [0..6]; I2C port number
0aaaaaaa:	aaaaaa = [0..127]; I2C address of device
0000bbbb:	
0000cccc:	bbbbcccc = [0..255]; data to write

Upto 4 databytes (bbbbcccc) may be written using one message by adding extra pairs of 0000bbbb and 0000cccc.

Example:

240, 125, 0 {DEV}, 124 {I2C WRITE SINGLE}, 04 {port number}, 73 {address of device}, 0, 3 {data byte 3 = 0*16+3}, 247 (F0h, 7Dh, 00h, 7Ch, 49h, 00h, 03h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C WRITE SINGLE message identical to the I2C WRITE SINGLE command sent. If there

was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (ie. in the above case the response would be F0h, 00h, 7Bh, 04h, 49h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with a I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Command ID: I2C READ (127, 7Fh)

The I2C READ command can be used to transfer data from a I2C device. Unless the I2C device has no pull-up resistors it is not necessary to use the I2C PORT command before sending the I2C READ command. The command can be used in both host and stand-alone modes of operation.

The I2C READ command [BODY] is:

00000xxx: xxx = [0..6]; I2C port number
0aaaaaaa: aaaaaa = [0..127]; I2C address of device
0000bbbb:
0000cccc: bbbbcccc = [0..255]; register number of device
0ddddddd: ddddddd = [1..127]; number of bytes to read

Example:

240, 125, 0 {DEV}, 127 {I2C READ}, 04 {port number}, 56 {address of device}, 0, 0 {register 0 = 0*16+0}, 2 {number of bytes to read 2}, 247 (F0h, 7Dh, 00h, 7Fh, 38h, 00h, 00h, 01h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C READ message having the same address, the same register number followed by the number of bytes that was requested and the data bytes (according to the example above, the reply message could be: F0h, 7Dh, 00h, 7Fh, 04h, 38h, 00h, 00h, 02h, 0Ah, 0Bh, 04h, 07h, F7h thus the received bytes would be ABh and 47h). Each requested byte is split as two consecutive bytes having their upper 4 bits to zero and encoded as upper byte sent first then lower byte (ex. 47h would be sent as 04h 07h). If there was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (ie. in the above case the response would be F0h, 00h, 7Bh, 04h, 38h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with an I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Command ID: I2C READ SINGLE (122, 7Ah)

The I2C READ SINGLE command can be used to transfer data from a I2C device. It is similar to the I2C READ command but is used for I2C devices that do not use a register access byte. Unless the I2C device has no pull-up resistors it is not necessary to use the I2C PORT command before sending the I2C READ SINGLE command. The command can be used in both host and stand-alone modes of operation.

The I2C READ SINGLE command [BODY] is:

00000xxx: xxx = [0..6]; I2C port number

0aaaaaaa: aaaaaa = [0..127]; I2C address of device
0bbbbbbb: bbbbbbb = [1..127]; number of bytes to read

Example:

240, 125, 0 {DEV}, 122 {I2C READ SINGLE}, 02 {port number}, 73 {address of device}, 3 {number of bytes to read 3}, 247 (F0h, 7Dh, 00h, 7Ah, 49h, 03h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C READ SINGLE message having the same address followed by the number of bytes that was requested and the data bytes (according to the example above, the reply message could be: F0h, 7Dh, 00h, 7Ah, 02h, 49h, 03h, 00h, 07h, 0Eh, 0Fh, 08h, 00h, F7h thus the received bytes would be 07h, EFh and 80h). Each requested byte is split as two consecutive bytes having their upper 4 bits to zero and encoded as upper byte sent first then lower byte (ex. 47h would be sent as 04h 07h). If there was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (ie. in the above case the response would be F0h, 00h, 7Bh, 02h, 49h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with an I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Transmitted messages

Note that in the following listing the commands as echoed back by the USB-microDig (to allow multiple Max/MSP objects to work with a USB-microDig) are not included.

Message ID: STREAM DATA (0, 00h)

In host mode the USB–microDig sends analog sensor input values, after being activated through a STREAM command, in the STREAM DATA message.

The [BODY] of the STREAM DATA message contains a list of sensor values from all active (turned on) sensors, in ascending order. Inactive sensors are not included in the list. Both 7-bit and 10-bit sensor values are packed together - 10-bit values are sent using two bytes, where the first byte contains the first 7 bits of the sensor value and the second byte contains the last 3 bits of the sensor value:

0yyyyyyy: (7-bit lo-res mode)

or

0yyyyyyy, 000zzz00: (10-bit hi-res mode)

yyyyyyy = [0..127]; the most significant bits (or a full data byte in lo-res mode) zzz = [0..7]; the 3 least significant bits (used in 10-bit hi-res mode only)

Example:

The USB-microDig has sensor input 1 turned on in lo-res mode (7-bit), sensor input 5 turned on in hi-res mode (10-bit), and sensor input 8 turned on in lo -res mode (7-bit). All other sensor inputs are turned off. The USB-microDig acquires the value 100 (sensor input 1), 1000 (sensor input 5), and 21 on (sensor input 8). The USB-microDig sends the following message:

240, 125, 0 {DEV}, 0 {STREAM DATA}, 100 {yyyyyyy of sensor input 1}, 125 {yyyyyyy of sensor input 5}, 0 {zzz of sensor input 5}, 21 {yyyyyyy of sensor input 8}, 247 (F0h, 7Dh, 00h, 00h, 64h, 7Dh, 00h, 15h, F7h)

The sampled value from sensor input 5 is 125 * 8 (i.e. 3 bit shift) + 0 = 1000

Message ID : SAMPLE DATA (4, 04h)

The SAMPLE DATA message contains a single snapshot of a sensor connected to a sensor input, whether the USB-microDig is in host or in stand-alone mode.

The [BODY] of the SAMPLE DATA message is composed of two or three bytes - the sensor input number, the first data byte, and, if the sensor input is set to 10-bit hi-res mode, the second data byte.

The [BODY] of the SAMPLE DATA message:

> 00000xxx, 0yyyyyyy: (7-bit lo-res mode)
> or
> 00000xxx, 0yyyyyyy, 000zzz00: (10-bit hi-res mode)
>
> xxx = [0..7]; sensor input number, where the first sensor input number = 0, and the last (8th) sensor input number = 7
>
> yyyyyyy = [0.127]; the most significant bits (or a full data byte in lo-res mode) zzz = [0..7]; the 3 least significant bits (used in 10-bit hi-res mode only)

Example:

When sampling sensor input 8 (provided it is turned off) and the sensor input is set to lo-res mode, the following message is received:

240, 125, 0 {DEV}, 4 {SAMPLE DATA}, 7 {xxx}, 64 {yyyyyyy}, 247 (F0h, 7Dh, 00h, 04h, 07h, 40h, F7h)

The sampled value from sensor input 8 is 64.

When sampling sensor input 8 (provided it is turned off) and the sensor input is set to hi-res mode, the following message is received:

240, 125, 0 {DEV}, 4 {SAMPLE DATA}, 7 {xxx}, 10 {yyyyyyy}, 10 {zzz}, 247 (F0h, 7Dh, 00h, 04h, 07h, 0Ah, 0Ah, F7h)

The sampled value from sensor input 8 is 10 * 8 (i.e. 3 bit shift) + 10 = 90.

Message ID: I2C DATA (10, 0Bh)

In host mode the USB–microDig sends digital (I2C) sensor values, after being activated through a FUNCTION command, in the I2C DATA message.

The [BODY] of the I2C DATA message contains a list of sensor values as specified in the function's method. 8-bit values are sent as two nibbles inside a byte each:

00000yyy:	yyy = [0..7]; sensor input number, where the first input number = 0 and the last (8th) input number = 7 yyy = [0..6]; I2C port number, where the first port number = 0 and the last (7th) port number = 6
0ddddddd:	ddddddd = [0-127]; I2C device address
0nnnnnnn:	nnnnnnn = [0-127]; function number
0ppppppp:	ppppppp = [0-127]; function method or parameter
0000dddd:	
0000eeee:	ddddeeee = [0..255]; data (number of bytes as per function specification)

Example:

The USB-microDig has function Orient3D, method 1, enabled an Orient3D sensor with device address 96 connected to I2C port 1. The USB-microDig acquires the value 90 (heading east), 45 (pitch down), and -45 (roll to the right). The USB-microDig sends the following message in host mode:

> 240, 125, 0 {DEV}, 10 {I2C DATA}, 0 {I2C port}, 96 (address of device}, 125 {Orient3D}, 1 {heading/pitch/roll}, 0 {dddd of heading 1st byte }, 10 {eeee of heading 1st byte}, 8 {dddd of heading 2nd byte}, 12 {eeee of heading 2nd byte}, 2 {dddd of pitch}, 0 {eeee of pitch}, 13 {dddd of roll}, 15 {eeee of roll}, 247 (F0h, 7Dh, 00h, 0Bh, 00h, 64h, 7Dh, 01h, 00h, 0Ah, 08h, 0Ch, 02h, 00h, 0Dh, 0Fh, F7h)

The heading value from the Orient3D is (0 * 16 (ie. 4-bit shift) + 10) * 256 (ie. 8-bit shift) + 8 * 16 (i.e. 4-bit shift) + 12 = 2700, corresponding to 270.0 degrees.
The pitch value from the Orient3D is 2 * 16 (i.e. 4-bit shift) + 0 = 32 unsigned, so that the two's complement signed value is also 32, corresponding to 32 * 359 / 255 = 45 degrees.
The roll value from the Orient3D is 13 * 16 (i.e. 4-bit shift) + 15 = 223 unsigned, so that the two's complement signed value is 223 – 256 = -32. Corresponding to -32 * 359 / 255 = -45 degrees.

Message ID: I2C PORT (125, 7Dh)

The I2C PORT message indicates if the port was opened by the I2C PORT command. The [BODY] of the I2C PORT message indicates the port I2C port number, whether it has been opened and whether its pull-up resistors have been enabled. If the USB-microDig is configured to stream data from a digital (I2C) sensor, ie. there are repeated attempts to open the port, this message is only sent once for the port and would only be sent again if the port still can't be opened after the port was turned off and on again with the STREAM command.

The I2C PORT message [BODY] is:

0xp00yyy:	x = 1; port enabled x = 0; all I2C ports disabled yyy = [0..6]; I2C port number, where port number = 0 corresponds with inputs 1-2 p = 1; internal pull-up resistors enabled p = 0; internal pull-up resistors disabled

Example:

240, 125, 0 {DEV}, 125 {I2C PORT}, 67 { x = 1, yyy = 3, p = 0}, 247 (F0h, 7Dh, 00h, 7Fh, 03h, F7h)

This message indicates that port 3 (inputs 4-5), without its internal pull-up resistors enabled, was opened for I2C communications.

Message ID: I2C NO RESPONSE (123, 7Bh)

The I2C NO RESPONSE message indicates that the I2C device was unresponsive, ie. there was no acknowledgement from the device within 100 ms, eg. caused by a wrong address or the device not being present. If the USB-microDig is configured to stream data from a digital (I2C) sensor, ie. a response is repeatedly requested from the I2C sensor, this message is only sent once for the port and would only be sent again if the unresponsiveness re-occurred after the port was turned off and on again with the STREAM command or after a response was obtained.

The I2C NO RESPONSE message [BODY] is:

00000xxx: xxx = [0..6]; I2C port number
0aaaaaaa: aaaaaa = [0..127]; I2C address of device

Example:

240, 125, 0 {DEV}, 123 {I2C NO RESPONSE}, 04 {port number}, 96 {address of device}, 247 (F0h, 7Dh, 00h, 7Fh, 04h, 60h, F7h) This message indicates that device 96 on port 4 (inputs 5-6) was unresponsive.

Message ID: I2C WRITE (126, 7Eh)

The I2C WRITE message is a copied reply from the I2C WRITE command. This message indicates that the I2C device received and acknowledged the I2C WRITE command sent to it's I2C address. The [BODY] of the I2C WRITE message indicates the I2C address of the device, the register number and the data bytes transfered to the device.

The I2C WRITE message [BODY] is:

00000xxx: xxx = [0..6]; I2C port number
0aaaaaaa: aaaaaa = [0..127]; I2C address of device
0000bbbb:
0000cccc: bbbbcccc = [0..255]; register number of device
0000dddd:
0000eeee: ddddeeee = [0..255]; data to write to register bbbbcccc

Example:

240, 125, 0 {DEV}, 126 {I2C WRITE}, 05 {port number}, 56 {address of device}, 0, 3 {register 3 = 0*16+3}, 1, 2 {data byte 18 = 1*16+2}, 247 (F0h, 7Dh, 00h, 7Eh, 05h, 38h, 00h, 03h, 01h, 02h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C WRITE message identical to the I2C WRITE command sent. If there was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, 05h, 38h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with a I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Message ID: I2C READ (127, 7Fh)

The I2C READ message is a reply from the I2C READ command. This message indicates that the I2C device supplied data according to the I2C READ command. The [BODY] of the I2C READ message indicates the I2C address of the device, the register number, the number of bytes transfered from the device and the bytes of data.

The I2C READ message [BODY] is:

00000xxx:	xxx = [0..6]; I2C port number
0aaaaaaa:	aaaaaa = [0..127]; I2C address of device
0000bbbb:	bbbbcccc = [0..255]; register number of device
0000cccc:	
0ddddddd:	ddddddd = [1..127]; number of bytes read
0000eeee:	
0000ffff:	eeeeffff = [0..255]; data byte #1
0000gggg:	
0000hhhh:	gggghhhh = [0..255]; data byte #2 if more than 1 byte requested

Example:

240, 125, 0 {DEV}, 127 {I2C READ}, 03 {port number}, 56 {address of device}, 0, 0 {register 0 = 0*16+0}, 2 {number of bytes read 2}, 10, 11 {data byte #1 171 = 10*16+11}, 4, 7 {data byte #2 71 = 4*16+7}, 247 (F0h, 7Dh, 00h, 7Fh, 03h, 38h, 00h, 00h, 01h, 0Ah, 0Bh, 04h, 07h, F7h)

If the device responded to the provided I2C address “aaaaaaa” (I2C devices always acknowledge when it’s address is called) then the USB-microDig will respond with a I2C READ message identical to the I2C READ command sent. If there was no acknowledgement from the device within 100 ms (eg. wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, 05h, 38h, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond with a I2C PORT “port not opened” message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested port).

Sensor Functions

This section describes the sensor-specific algorithms that enable the USB-microDig to process analog sensor signals and/or to communicate with digital (I2C) sensors. The functions allow digital sensors to be used in stand-alone mode, mapped to channel voice MIDI messages, as well as in host mode sent out as I2C DATA messages. The functions allow analog sensor signals to be linearized or converted before being mapped to channel voice MIDI values in stand-alone mode or sent out as STREAM messages in host mode.

Analog Sensors

Function ID: LOOKUP (1, 01h)

Each value captured by an analog sensor input of the USB-microDig can be translated to another value by using a lookup table. This allows for linearization of sensor values or could do exactly the reverse by translating the sensor values in a non-linear fashion. The lookup table entries can be set with the TABLE ENTRY command, and read using the SEND TABLE command. Each table contains 1024 14-bit values stored as 2048 8-bit bytes and sent as 2048 7-bit bytes. Even though the non-volatile memory can store 16-bit values, only 14-bits are made accessible with the TABLE ENTRY command and since analog sensor values have 10-bit resolution (ie. 0..1023 range) internally in the USB-microDig, it's only useful to store 10-bit values. Hence the table entries should be in the range 0..1023 to span the full range of a channel voice MIDI message value, whether it is 0..127 such as in control-change MIDI messages or 0..16383 in pitch-bend messages.

Tables can be made using software downloadable from our website. The software includes the following tables:

1: BendMicro;
2: BendMini;
3: BendShort;
4: ReachClose;
5: ReachFar;
6: TapTile;
7: Touch;
8: TouchMicro-03;
9: TouchMicro-05;
10: TouchMicro-10;
11: TouchMini;
12: TouchStrip;

Example:

In order to use the ReachClose sensor on the 1st input with the ReachClose lookup table, set the 1th sensor input to use lookup table 4, the following message should be sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {input number}, 0 {device address (ignored)}, 1 {lookup method}, 4 {table number}, 247 (F0h, 7Dh, 00h, 08h, 00h, 01h, 04h, F7h)

Function ID: TIME (2, 02h)

The timer function measures the time between two events defined by the peak and end-peak features defined in the EDIT CONFIG command. The following methods can be selected using the FUNCTION command's parameter values FP1:

FP1	0yyyyyyy:	yyyyyyy = [1..127]; Scale factor yyyyyyy = 1; time is output in ms yyyyyyy = 100; time is output in 10ths of a second

Example:

In order to output the time in ms of a tap on a Touch sensor connected to input 1 of the USB-microDig as MIDI controller 64 messages, the following messages are sent (see also the EDIT CONFIG command):

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {yyy = 0}, 0 {FP2}, 2 {time function}, 1 {ms}, 247 (F0h, 7Dh, 00h, 08h, 00h, 3Ch, 02h, 01h, F7h)

240, 125, 0 {DEV}, 106 {EDIT CONFIG}, 1 {configuration number}, 0 {input}, 48 {control-change message}, 64 {controller number}, 34 {peak detection with end notification}, 64 {threshold}, 127 {top}, 0 {noisegate}, 0 {time window}, 247 (F0h, 7Dh, 00h, 6Ah, 01h, 00h, 30h, 40h, 22h, 40h, 7Fh, 00h, 00h, F7h)

The threshold value may need adjustment to properly capture the heartbeat signal peaks.

Function ID: FREQUENCY (3, 03h)

The frequency function measures either the frequency of successive peaks or the frequency calculated from the time between peak and end-peak events defined in the EDIT CONFIG command. The following methods can be selected using the FUNCTION command's parameter values FP1:

FP1	0yyyyyyy:	yyyyyyy = [1..127]; Scale factor yyyyyyy = 1; Frequency is output as Hz yyyyyyy = 60; Frequency is output as BPM

The output range is divided by 1000 times the scale factor, so that if the scale factor is set to 1 and the frequency is output as pitch-bend messages, the frequency range is approx. 0 Hz – 16 KHz (Hertz). With the scale factor set to 60, and the frequency output as control-change messages, the frequency range is 0 – 127 BPM (beats per minute).

Example:

In order to output the frequency in BPM taps on a Touch sensor connected to input 1 of the USB-microDig as MIDI controller 64 messages, the following messages are sent (see also the EDIT CONFIG command):

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {yyy = 0}, 0 {FP2}, 3 {frequency function}, 60 {BPM}, 247 (F0h, 7Dh, 00h, 08h, 00h, 00h, 03h, 3Ch, F7h)

240, 125, 0 {DEV}, 106 {EDIT CONFIG}, 1 {configuration number}, 0 {input}, 48 {control-change message}, 64 {controller number}, 2 {peak detection}, 64 {threshold}, 127 {top}, 0 {noisegate}, 0 {time window}, 247 (F0h, 7Dh, 00h, 6Ah, 02h, 00h, 30h, 40h, 22h, 40h, 7Fh, 00h, 00h, F7h)

The threshold value may need adjustment to properly capture the heartbeat signal peaks.

Function ID: ABSOLUTE (4, 04h)

The absolute function provides a way to convert signals that vary around a center value (such as from accelerometers and physiological sensors) to a value from zero. The following methods can be selected using the FUNCTION command's parameter values FP1 and FP2:

FP1	0xxxxxxx:	xxxxxxx = [0..127]; Offset high bits
FP2	00000yyy:	yyy = [0..7]; Offset low bits

The offset is calculated as a 10-bit value represented by xxxxxxxyyy.

Example:

In order to output the value generated by a sudden motion of a GForce3D-6 sensor as a MIDI controller 64 message, the following messages are sent (see also the EDIT CONFIG command):

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {input}, 0 {offset least significant bits}, 4 {absolute function}, 64 {offset MSB}, 247 (F0h, 7Dh, 00h, 08h, 00h, 00h, 04h, 40h, F7h)

240, 125, 0 {DEV}, 106 {EDIT CONFIG}, 1 {configuration number}, 0 {input}, 48 {control-change message}, 64 {controller number}, 34 {peak detection with end notification}, 1 {threshold}, 64 {top}, 0 {noisegate}, 0 {time window}, 247 (F0h, 7Dh, 00h, 6Ah, 01h, 00h, 30h, 40h, 22h, 40h, 7Fh, 00h, 00h, F7h)

The threshold value may need adjustment to properly capture the signal peaks.

Function ID: BIOBEAT (10, 0Ah)

The BioBeat function measures the frequency of successive peaks as defined in the EDIT CONFIG command. The following methods can be selected using the FUNCTION command's parameter values FP1 and FP2:

FP1	0xxxxxxx:	xxxxxxx: [0..127]; Offset
FP2	0m00yyyy:	yyyy: [0..15]; Smoothing m = 0; monitor off m = 1; monitor on

When the monitor is enabled, the processed sensor value before peak detection is output as MIDI note-on or control-change messages, with the number being one higher than the note or controller number set by the EDIT CONFIG command.

Example:

In order to output the frequency in BPM of a BioVolt sensor connected to input 1 of the USB-microDig (in stand-alone mode with USB-microMIDICable and battery-powered so as to prevent electrical shock) as MIDI controller 64 messages, the following messages are sent (see also the EDIT CONFIG command):

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {yyy = 0}, 100 {smoothing}, 10 {biobeat function}, 64 {offset}, 247 (F0h, 7Dh, 00h, 08h, 00h, 64h, 0Ah, 40h, F7h)

240, 125, 0 {DEV}, 106 {EDIT CONFIG}, 1 {configuration number}, 0 {input}, 48 {control-change message}, 64 {controller number}, 34 {peak detection and smoothing}, 64 {threshold}, 127 {top}, 0 {noisegate}, 0 {time window}, 247 (F0h, 7Dh, 00h, 6Ah, 02h, 00h, 30h, 40h, 22h, 40h, 7Fh, 00h, 00h, F7h)

The threshold value may need adjustment to properly capture the heartbeat signal peaks.

Digital (I2C) Sensors

Function ID: ORIENT3D (125, 7Dh)

The Orient3D function enables output of the Orient3D sensor data in both host and stand-alone modes using a number of methods. In host mode the sensor data is output as an I2C DATA message. In stand-alone mode the sensor data is output according to the stand-alone mode settings of the input number that equals the I2C port number. Since the Orient3D has multiple output parameters the stand-alone mode settings are incremented by one for each extra output parameter. The following methods can be selected using the FUNCTION command's method/parameter value FP:

Orient3D v3.0

FP

1: Heading [0..3599]; an unsigned 16-bit value, where 3599 represents 359.9 degrees, sent as two 8-bit bytes (MSB, LSB)
Pitch [-85..85]; a signed 8 bit value, where 255 represents 359 degrees, sent as one 8-bit byte
Roll [-85..85]; a signed 8-bit value, where 255 represents 359 degrees, sent as one 8-bit byte

2: Magnetic Field Strength X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

3: Acceleration X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Acceleration Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

Acceleration Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

9: Magnetic Field Strength X, Y, Z, Acceleration X, Y, Z, in the same formats as above (host mode only)

Orient3D v3.1

FP

11: Heading [0..3599]; an unsigned 16-bit value, where 3599 represents 359.9 degrees, sent as two 8-bit bytes (MSB, LSB)
Pitch [-90..90]; a signed 8 bit value, where 255 represents 359 degrees, sent as one 8-bit byte
Roll [-90..90]; a signed 8-bit value, where 255 represents 359 degrees, sent as one 8-bit byte

12: Magnetic Field Strength X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

13: Acceleration X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Acceleration Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Acceleration Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

14: Gyro X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Gyro Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Gyro Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes

19: Magnetic Field Strength X, Y, Z, Acceleration X, Y, Z,
Gyro X, Y, Z, in the same formats as above (host mode only)

Example:

In order to enable the Orient3D function to get the heading, pitch and roll data from an Orient3D sensor with I2C device address 96 (default Orient3D I2C device address) connected to the 1st I2C port, the following message is sent (see also the FUNCTION command):

240, 125, 0 {DEV}, 8 {FUNCTION}, 0 {yyy = 0}, 96 {device address}, 125 {Orient3D function}, 1 {heading/pitch/roll}, 247 (F0h, 7Dh, 00h, 08h, 00h, 60h, 7Dh, 01h, F7h)

The USB-microDig acquires the value 90 (heading east), 45 (pitch down), and -45 (roll to the right). The USB-microDig sends the following message in host mode:

240, 125, 0 {DEV}, 10 {I2C DATA}, 0 {I2C port}, 96 (address of device}, 125 {Orient3D}, 1 {heading/pitch/roll}, 0 {dddd of heading 1st byte }, 10 {eeee of heading 1st byte}, 8 {dddd of heading 2nd byte}, 12 {eeee of heading 2nd byte}, 2 {dddd of pitch}, 0 {eeee of pitch}, 13 {dddd of roll}, 15 {eeee of roll}, 247 (F0h, 7Dh, 00h, 0Bh, 00h, 64h, 7Dh, 01h, 00h, 0Ah, 08h, 0Ch, 02h, 00h, 0Dh, 0Fh, F7h)

The heading value from the Orient3D is (0 * 16 (ie. 4-bit shift) + 10) * 256 (ie. 8-bit shift) + 8 * 16 (i.e. 4-bit shift) + 12 = 2700, corresponding to 270.0 degrees.
The pitch value from the Orient3D is 2 * 16 (i.e. 4-bit shift) + 0 = 32 unsigned, so that the two's complement signed value is also 32, corresponding to 32 * 359 / 255 = 45 degrees.
The roll value from the Orient3D is 13 * 16 (i.e. 4-bit shift) + 15 = 223 unsigned, so that the two's complement signed value is 223 – 256 = -32. Corresponding to -32 * 359 / 255 = -45 degrees.

In host mode raw sensor data will be output immediately in I2C DATA messages.

In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. For example to output the Orient3D data as a MIDI controller 64 messages, the following message is sent:

> 240, 125, 0 {DEV}, 106 {EDIT CONFIG}, 1 {configuration number}, 0 {input}, 48 {control-change message}, 64 {controller number}, 1 {continuous signal processing}, 0 {threshold}, 127 {top}, 0 {noisegate}, 0 {time window}, 247 (F0h, 7Dh, 00h, 6Ah, 01h, 00h, 30h, 40h, 01h, 00h, 7Fh, 00h, 00h, F7h)

The sensor parameter's raw values as output in a I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0-127 (eg. control-change message) or 0-16383 (pitch-bend message).

Other settings of the Orient3D sensor can be effectuated by using I2C commands such as I2C WRITE and I2C READ, which can be sent in both host and stand-alone mode.

Function ID: MOVEAROUND (124, 7Ch)

The MoveAround function enables output of the MoveAround sensor data in both host and stand-alone mode. In host mode the sensor data is output as an I2C DATA message. In stand-alone mode the sensor data is output according to the stand-alone mode settings of the input number that equals the I2C port number. Since the MoveAround has multiple output parameters the stand-alone mode settings are incremented by one for each extra output parameter. There are no methods to select, ie. the FUNCTION command's method/parameter value is ignored:

MoveAround v1.0

FP

0..127: Ambient Temperature; an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 1); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 2); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 3); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 4); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 5); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 6); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 7); an unsigned 8-bit value sent as one 8-bit byte
Temperature (pixel 8); an unsigned 8-bit value sent as one 8-bit byte

Each 8-bit value represents the temperature value [4..100] in degrees Celsius.

Example:

In order to enable the MoveAround function to get the temperature data from a MoveAround sensor with I2C device address 104 (default MoveAround I2C device address) connected to the 7th I2C port, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 6 {yyy = 6}, 104 {device address}, 124 {MoveAround function}, 0 {all temperatures}, 247 (F0h, 7Dh, 00h, 08h, 06h, 68h, 7Ch, 00h, F7h)

In host mode raw sensor data will be output immediately in I2C DATA messages. In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. The sensor parameter's raw values as output in a I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0-127 (eg. control-change message) or 0-16383 (pitch-bend message).

Other settings of the MoveAround sensor can be effectuated by using I2C commands such as I2C WRITE and I2C READ.

Function ID: MOVEALONG (123, 7Bh)

The MoveAlong function enables output of the MoveAlong sensor data in both host and stand-alone mode. In host mode the sensor data is output as an I2C DATA message. In stand-alone mode the sensor data is output according to the stand-alone mode settings of the input number that equals the I2C port number. There are no methods to select, ie. the FUNCTION command's method/parameter value is ignored:

MoveAlong v1.0

FP

0..127: Angular Velocity [-512..511]; a signed two's complement 16-bit value, representing degrees/second, sent as two 8-bit bytes (MSB, LSB)

Example:

In order to enable the MoveAlong function to get the angular velocity data from an MoveAlong sensor with I2C device address 0 (default MoveAlong I2C device address) connected to the 7th I2C port, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 6 {yyy = 6}, 1 {device address}, 122 {MoveAlong function}, 0 {angular velocity method}, 247 (F0h, 7Dh, 00h, 08h, 06h, 01h, 7Bh, 00h, F7h)

In host mode raw sensor data will be output immediately in I2C DATA messages. In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. The sensor parameter's raw values as output in a I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0-127 (eg. control-change message) or 0-16383 (pitch-bend message).

Other settings of the MoveAlong sensor can be effectuated by using I2C commands such as I2C WRITE and I2C READ.

Function ID: MAGNETIC3D (122, 7Ah)

The Magnetic3D function enables output of the Magnetic3D sensor data in both host and stand-alone mode. In host mode the sensor data is output as an I2C DATA message. In stand-alone mode the sensor data is output according to the stand-alone mode settings of the input number that equals the I2C port number. Since the Magnetic3D has multiple output parameters the stand-alone mode settings are incremented by one for each extra output parameter. There are no methods to select, ie. the FUNCTION command's method/parameter value is ignored:

Magnetic3D v1.0

FP

0..127: Magnetic Field Strength X-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Y-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)
Magnetic Field Strength Z-axis; a signed two's complement 16-bit value sent as two 8-bit bytes (MSB, LSB)

Each 16-bit value represents a magnetic field strength that is scaled using a factor that can be set with a separate I2C WRITE command, see for more details the Magnetic3D product page.

Example:

In order to enable the Magnetic3D function to get the magnetic field strength X-, Y- and Z-axis data from an Magnetic3D sensor with I2C device address 30 (default Magnetic3D I2C device address) connected to the 7th I2C port, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 6 {yyy = 6}, 30 {device address}, 122 {Magnetic3D function}, 0 {X-/Y-/Z-axis}, 247 (F0h, 7Dh, 00h, 08h, 06h, 1Eh, 7Ah, 00h, F7h)

In host mode raw sensor data will be output immediately in I2C DATA messages. In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. The sensor parameter's raw values as output in a I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0-127 (eg. control-change message) or 0-16383 (pitch-bend message).

Other settings of the Magnetic3D sensor can be effectuated by using I2C commands such as I2C WRITE and I2C READ.

Function ID: MOIST (121, 79h)

The Moist function enables output of the Moist sensor data in both host and stand-alone mode. In host mode the sensor data is output as an I2C DATA message. In stand-alone mode the sensor data is output according to the stand-alone mode settings of the input number that equals the I2C port number. Since the Moist has multiple output parameters the stand-alone mode settings are incremented by one for each extra output parameter. There are no methods to select, ie. the FUNCTION command's method/parameter value is ignored:

Moist v1.0

FP

1: Soil Moisture [0..512]; an unsigned 16-bit value sent as two 8-bit bytes (MSB, LSB)
2: Illumination [0..65536]; an unsigned 16-bit value sent as two 8-bit bytes (MSB, LSB)
3: Temperature [0..512]; an unsigned 16-bit value, where 512 represents 51.2 degrees Celsius, sent as two 8-bit bytes (MSB, LSB)
4: Soil Moisture, Illumination, Temperature; unsigned 16-bit values each sent as two 8-bit bytes (MSB, LSB)

There are no calibration factors available for the Soil Moisture and Illumination values, ie. it is not known what is represented by each value.

Example:

In order to enable the Moist function to get all sensor data from a Moist sensor with I2C device address 32 (default Moist I2C device address) connected to the 7th I2C port, the following message is sent:

240, 125, 0 {DEV}, 8 {FUNCTION}, 6 {yyy = 6}, 32 {device address}, 121 {Moist function}, 4 {all parameters}, 247 (F0h, 7Dh, 00h, 08h, 06h, 20h, 79h, 04h, F7h)

In host mode raw sensor data will be output immediately in I2C DATA messages. In stand-alone mode a suitable configuration has to be set using the EDIT CONFIG command before any MIDI messages will be output. The sensor parameter's raw values as output in a I2C DATA message are offset and scaled so as to fit in the channel voice MIDI value range of either 0-127 (eg. control-change message) or 0-16383 (pitch-bend message).

Other settings of the Moist sensor can be effectuated by using I2C commands such as I2C WRITE and I2C READ.
