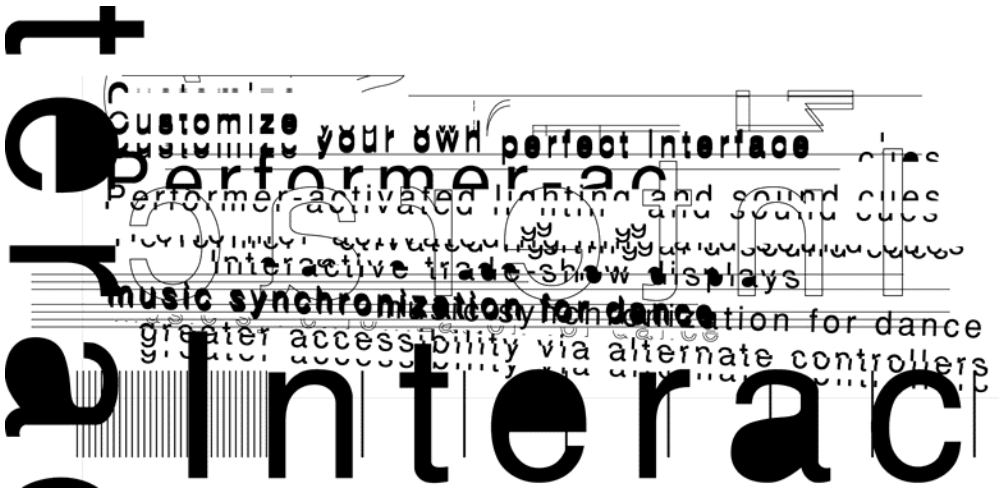
USB-microDig communication protocol v7.0

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

24 April 2008



I-CubeX USB-microDig communication protocol v7.0

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 include 8 digital outputs that can be used to control actuators using the top (4th) row on the sensor input connector. The outputs are controlled the same way as the I-CubeX digitizer either by MIDI messages such as note ON, note OFF, Key pressure, Control change or by system exclusive messages (you can use our free ocube Max/MSP plugin).

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 of 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 2000/XP/Server 2003/Vista or OSX). You can then establish a connection with the USB-microDig by using our free 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, data flow ON.

The USB-microDig is self powered and provides 5V power to the connected sensors/actuators. It is registered as a device that use 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 bridging application that makes the serial port data available at a MIDI port. 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 SPP bridging application to MIDI.

This document describes the I-CubeX USB-microDig communication protocol v7.0. 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.

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

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, setting the mode of operation (stand-alone or host mode), setting the device ID, setting MIDI thru operation and obtaining version information. These commands have the same effect both in stand-alone and host mode. This section also describes system status messages.

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 a visual feedback when power is received from the USB connector. When the USB-microDig COM port is opened (using our free I-CubeX editor or a customer provided COM port application) the POWER LED will be ON constantly. The USB-microDig can detect when the COM port is open by the use of hardware dataflow (RTS/CTS). If the software application you are using doesn't have hardware dataflow (such as the Max/MSP *serial* object) then the COM port can be open but the POWER LED on the USB-microDig will keep on flashing until it receives serial data (that your software application sent). Then the POWER LED will stay constantly ON even after the COM port is closed. If you have a software application that has hardware dataflow, when the USB-microDig COM port is closed (by default when the USB-microDig is being connected) the POWER LED will blink again.
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.

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. 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 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). 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. The RESET command is always executed following a power-up. 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:

240, 125, 0 {DEV}, 34 {RESET}, 247 (F0h, 7Dh, 00h, 22h, F7h)
is same as sending this single byte: 255 (FFh)

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

Mode of operation:	0; host mode
MIDI thru activation status:	retrieved from non-volatile memory (see MIDI THRU command)
System exclusive ID:	retrieved from non-volatile memory (see SET ID command)
Sensor sampling interval:	100 ms
Sensor input sampling resolution:	7 bit
Sensor input activation status:	0; off
Sensor input muting:	0; off

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

Mode of operation:	1; stand-alone mode
MIDI thru activation status:	retrieved from non-volatile memory (see MIDI THRU command)
System exclusive ID:	retrieved from non-volatile memory (see SET ID command)
Sensor input muting:	0; off
Other sensor input settings:	according to settings stored in non-volatile memory (see EDIT CONFIG 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 modes. The USB-microDig can function in host mode, or stand-alone mode.

Stand-alone mode allows the USB-microDig to be used with its internal signal processing so that the host computer can directly use the calculated end result. All processing and mapping happens in the USB-microDig itself, so it can be patched to a MIDI software such as a VST synthesizers or such. This mode allows sensor inputs to be mapped to channel voice MIDI messages only (ie. no host mode type system exclusive MIDI messages are used for outputting sensor values). 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 be connected with a software that will control and store the USB-microDig settings. In host mode unprocessed 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). 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 the section on host mode commands.

When switching the operation mode, 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 factory 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 mode of operation. 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 factory 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 THRU (93, 5Dh)

This command sets the software MIDI thru operation of the USB-microDig. Note that other MIDI devices with a MIDI thru connector implement MIDI thru in hardware (resulting in much less delay). When MIDI thru is on, all MIDI messages that are not applicable to the USB-microDig are sent out the MIDI output. Applicable messages are the system exclusive messages with the manufacturer ID used by Infusion Systems as well as the Device ID used by the USB-microDig. In addition, when operating in stand-alone mode, other applicable messages are the channel voice messages for the MIDI channel the USB-microDig is set to using the EDIT CONFIG command. The default factory value is off (no MIDI thru). Upon completion of the command the USB-microDig sends out the same command.

The [BODY] of the MIDI THRU command is:

0xxxxxxx: xxxxxxx = 0; MIDI thru off
xxxxxxx > 0; MIDI thru on

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

This command sets the MIDI running status feature of the USB-microDig.
When this is set to ON (xxxxxxx > 0) 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 set to OFF (xxxxxxx = 0) 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 off
	xxxxxxx > 0; MIDI running status on

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}, 70 {x}, 70 {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 70 / 10 = 7.0. The hardware board version number is 70 / 10 + 0 / 1000 = 7.00. 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

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, for stand-alone mode. This mode allows the USB-microDig to be used without the need of software to process the sensor data because all settings are stored in non-volatile memory (EEPROM). All processing and mapping happens in the USB-microDig itself, so it can be directly connected to a MIDI software synthesizers once it has been programmed using the I-CubeX editor (free to download from I-cubeX.com). This mode allows sensor inputs to be mapped to channel voice MIDI messages only (ie. no host mode type system exclusive MIDI messages are used for outputting sensor values). 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.

Programming commands

Command ID: STREAM (1, 01h)

Each sensor input of the USB-microDig can be turned on or off by using the STREAM command if a signal analysis method has been selected (impulse and/or continuous signal analysis, see EDIT CONFIG command). If no signal analysis method has been selected the STREAM command has no effect. In stand-alone mode the activation status of the sensor input is stored in non-volatile memory as well as volatile memory for immediate use, while in host mode the activation status is only stored in volatile memory. 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 each time the mapped sensor value changes. 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 analysis methods is activated. The factory default value of the sensor activation status is 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 (8th) sensor input number = 7

Example:

In order to turn ON the 3rd 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: INTERVAL (3, 03h)

The USB-microDig's global sampling interval can be set to speed up or slow down data acquisition. The interval is set in milliseconds with 1 ms being the lowest (ie. highest rate), and 16383 ms (about 16 seconds) being the highest (ie. lowest rate). When the INTERVAL command is sent in stand-alone mode the sample interval will be stored in non-volatile memory as well as in volatile memory for immediate use, while in host mode it will be stored in volatile memory only. The factory default setting of the sampling interval is 100ms. After a RESET in stand-alone mode, the sampling interval is retrieved from non-volatile memory while in host mode it is set to the factory default value of 100ms. Upon completion of the command the same command is sent out by the USB-microDig.

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.

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: To set the sampling interval to 1 second (1000 ms), the following command is sent:
240, 125, 0 {DEV}, 3 {INTV}, 7 {x}, 104 {y}, 247 (F0h, 7Dh, 00h, 03h, 07h, 68h, F7h)
The sample interval is 7 * 128 + 104 = 1000

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, factory default = "U" (55h)
0bbbbbb: bbbbbb = [0..127]; 2nd character, factory default = "S" (53h)
0ccccccc: ccccccc = [0..127]; 3rd character, factory default = "B" (42h)
0ddddddd: ddddddd = [0..127]; 4th character, factory default = "-" (2Dh)
0eeeeeee: eeeeeee = [0..127]; 5th character, factory default = "u" (75h)
0fffffff: fffffff = [0..127]; 6th character, factory default = "D" (44h)
0ggggggg: ggggggg = [0..127]; 7th character, factory default = "i" (69h)
0hhhhhhh: hhhhhhh = [0..127]; 8th character, factory 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 factory 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

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

MIDI thru:	0; off
MIDI running status	1; on
System exclusive ID:	0
Mode of operation:	1; stand-alone
Configuration name:	“USB-uDig”
Sensor sampling rate:	100 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 analysis:	0; off
Sensor input continuous signal analysis:	0; off
Sensor input threshold:	0
Sensor input ceiling:	127
Sensor input noise gate:	0; off
Sensor input time window:	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 initialisation:	1; on
Actuator output MIDI value threshold:	1

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 analysis 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 signal averaging (0 = off, 1 = on) i = [0,1]; impulse signal analysis (0 = off, 1 = on) j = [0,1]; continuous signal analysis (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} qqqq = [0..15]; time window

Factory 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 note number mapping {ddddddd}:	sensor input nr {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 averaging {h}:	0; off
Sensor input impulse signal analysis {i}:	0; off
Sensor input continuous signal analysis {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 {qqqq}:	0; off

Mapping sensor inputs

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

The impulse signal analysis 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} in which the time window {qqqq} is not used and which provides a way to output each impulse maximum/minimum (peak/dip) as a constant of value 16x{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} ({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 analysis settings are the activation status {j} which will also start sampling of the sensor input, the averaging setting {g} which enables calculation of the average of a number of sensor values, the time window setting {qqqq} within which {qqqq} + 1 sensor values are taken to calculate the average value, the differentiation setting {g} which enables calculation of the difference between the current and last output value.

Continuous signal analysis can be combined with impulse signal analysis 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 analysis 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 ({e} = 0) to avoid sending out MIDI value zero).

In both impulse and signal analysis 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} divided by 2 (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 analysis 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 analysis are active the sensor value's start and end are characterized using impulse signal analysis and represented as one of the channel voice MIDI messages listed above, while the signal between start and end is characterized using continuous signal analysis and always represented as the key-pressure value of a MIDI channel voice key-pressure (header A0h) message. Note that if averaging is activated, {qqqq} will also be used for determining the averaging window, ie. {qqqq} + 1 samples will be taken for peak/dip detection and subsequently {qqqq} + 1 samples will be taken for averaging.

For example, if impulse signal analysis 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 analysis 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 analysis is activated, MIDI messages calculated as defined by the signal analysis method 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.

Editing 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
0aaabbbb:	aaa = [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
	d,e,f,g = 1; toggle mode
0000hijk:	h,i,j,k are response mode bits for actuator outputs 4 (h) .. 1 (k)
	h,i,j,k = 0; trigger mode
	h,i,j,k = 1; toggle mode
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) .. 5 (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

Factory default settings for all actuator outputs are:

Actuator output MIDI mapping type {aaa}: 1; note-on
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}: 1; toggle mode
Actuator output initialisation {m,n,p,q,r,s,t,u}: 1; on
Actuator output MIDI value threshold {vvvvvvv}: 1

Mapping actuator outputs

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 {vvvvvvv} turns actuator output on and a message with velocity or data field 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 {vvvvvvv} toggles actuator output both on and off (a message with velocity or data field zero has no effect).

Base note is the note number or control number which trigger or toggle actuator output 1.

Running status is implemented in the MIDI input mapping when using stand-alone mode.

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.

Host 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, for host mode. This mode requires that the USB-microDig is connected with a host computer to control and store the USB-microDig settings. 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). Host mode allows the use of multiple USB-microDigs. When using Max/MSP software, it 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.

Programming commands

Command ID: STREAM (1, 01h)

Each sensor input of the USB-microDig can be turned on or off by using the STREAM command. Upon completion of the command the activation status of the sensor input is stored in volatile memory only, when in host mode, and in both volatile as well as non-volatile memory when in stand-alone memory. In host mode STREAM messages are output, while in stand-alone mode channel voice messages are output. After a RESET command in host mode each sensor input is turned off, while in stand-alone mode the activation status is retrieved from non-volatile memory.

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 (8th) sensor input number = 7

Example:

In order to turn the 7th sensor input ON, the following message is sent:

240, 125, 0 {DEV}, 1 {STREAM}, 70 {x = 1, yyy = 6}, 247 (F0h, 7Dh, 00h, 01h, 46h, F7h)

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

240, 125, 0 {DEV}, 1 {STREAM}, 6 {x = 0, yyy = 6}, 247 (F0h, 7Dh, 00h, 01h, 06h, 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 by default (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: 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 rate), and 16383ms (about 16 seconds) being the highest (ie. lowest 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 100ms, while in stand-alone mode it is retrieved from non-volatile memory. Upon completion of the command the USB-microDig sends out the same command.

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.

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: SAMPLE (4, 04h)

The SAMPLE command provides a snapshot of the data coming out of a single sensor input. In order to use SAMPLE for a sensor input, that sensor input *must* be turned off (it is redundant to sample a sensor input that is turned on since it is already being sampled continuously). 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: OUTPUT (48, 30h)

The OUTPUT command can be used to turn any of the 8 available actuator outputs on or off. It can be used in both modes of operation. After a RESET with the USB-microDig in host mode, all outputs are turned off. Upon completion of the command the digitizer sends out the same command.

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

0x000yyy: x = 1; on
x = 0; off
yyy = [0..7]; actuator output number, where the first actuator output number = 0, and the last (8th) actuator output number = 7

Example:

In order to turn the 2nd actuator output on, the following message is sent:
240, 125, 0 {DEV}, 48 {RES}, 65 {x = 1, yyy = 1}, 247 (F0h, 7Dh, 00h, 30h, 41h, F7h)

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

240, 125, 0 {DEV}, 48 {RES}, 1 {x = 0, yyy = 1}, 247 (F0h, 7Dh, 00h, 30h, 01h, F7h)

Command ID: SENSOR CURRENT OVERLOAD (111, 6Fh)

In host mode (or in stand alone mode) the host can send a request to the USB-microDig to get the sensor current overload status. This feature is useful to get the status of the voltage at the sensor input connector. A low voltage level status would mean that the sensors are no longer at 5V.
When sending the SENSOR CURRENT OVERLOAD command the USB-microDig will respond with a "normal level" reply (F0h 7Dh 00h 6Fh 00h F7h) if the power voltage level at the sensor connector is 5.0V. However, if the battery input voltage level is below then the "low voltage level" message will be replied (F0h 7Dh 00h 6Fh 01h F7h).

Note that the RED power LED will be dimmed if the voltage level on the battery is below 5.0V but the USB-microDig will work.

There is no [BODY] for the SENSOR CURRENT OVERLOAD command.

Example:

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

Command ID: I2C PORT SET (125, 7Dh)

In host mode only, the USB-microDig can be used to communicate with I2C devices. 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 need to use 2 adjacent inputs (input pair) for its DATA (DA) and CLOCK (CK) lines. Using the "I2C PORT SET" command you can set the I2C port 0 to 6 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 SET COMMAND with port number 6 (6^{th} pair from input pair 1-2). 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 (between 4.7K to 10K resistor) before the I2C PORT SET 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 SET command with a I2C PORT message having 127 (7Fh) instead of the selected port number (F0h 7Dh 00h 7Dh 7Fh F7h). If the port has pull-up resistors then the USB-microDig will reply with a connected port message (F0h 7Dh 00h 7Dh xx F7h, where xx is the requested channel). Any invalid port number (i.e. 7 or higher) will give a “CONFIGURATION SETTING ERROR STATUS” message (F0h 7Dh 00h 25h 5Ah F7h).

The USB-microDig can have up to four I2C ports connected (side to side port 0, 2, 4 and 6), however an I2C PORT SET command must be issued each time you want to access the port you want to use. For example if you have two I2C ports connected to the USB-microDig: port 0 (input pairs 1-2) and port 2 (input pairs 3-4). To access the device on port 0 you must first send the I2C PORT SET command with port 0 then use the I2C

READ or I2C WRITE commands. Once access on port 0 is done you can send the I2C PORT SET command with port 2 then use the I2C READ or I2C WRITE commands to access the other devices on this port.

A port can have as many devices as there are different I2C addresses. Since a device only responds when it's 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*4=512 devices!
The limit would however be the current consumption taken from the inputs (30mA max per input) of the USB-microDig.

The USB-microDig can only be used as an I2C master (no I2C slave mode).
The USB-microDig can both use analog sensor 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).

Command ID: I2C WRITE (126, 7Eh)

Once the USB-microDig has an opened I2C channel (see the I2C PORT SET command description) the I2C WRITE command can be used to transfer data to a I2C device.

The I2C WRITE command [BODY] is:

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}, 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 acknowledge from the device (wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond to the I2C WRITE command with a "CONFIGURATION SETTING ERROR STATUS" message (F0h 7Dh 00h 25h 5Ah F7h).

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.
Once the USB-microDig has an opened I2C channel (see the I2C PORT SET command description) the I2C WRITE SINGLE command can be used to transfer data to a I2C device.

The I2C WRITE SINGLE command [BODY] is:

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

Example:

240, 125, 0 {DEV}, 124 {I2C WRITE SINGLE}, 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 acknowledge from the device (wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond to the I2C WRITE SINGLE command with a “CONFIGURATION SETTING ERROR STATUS” message (F0h 7Dh 00h 25h 5Ah F7h).

Command ID: I2C READ (127, 7Fh)

Once the USB-microDig has an opened I2C channel (see the I2C PORT SET command description) the I2C READ command can be used to transfer data from a I2C device.

The I2C READ command [BODY] is:

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}, 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 with 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, 38h, 00h, 00h, 02h, 0Ah, 0Bh, 04h, 07h, F7h thus the received bytes would be ABh and 47h). Each requested bytes are 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 acknowledge from the device (wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond to the I2C WRITE command with a “CONFIGURATION SETTING ERROR STATUS” message (F0h 7Dh 00h 25h 5Ah F7h).

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

The I2C READ SINGLE command is similar to the I2C READ command but is used for

I2C devices that do not use a register access byte.
Once the USB-microDig has an opened I2C channel (see the I2C PORT SET command description) the I2C READ SINGLE command can be used to transfer data from a I2C device.

The I2C READ SINGLE command [BODY] is:

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}, 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 with 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, 49h, 03h, 00h, 07h, 0Eh, 0Fh, 08h, 00h, F7h thus the received bytes would be 07h, EFh and 80h). Each requested bytes are 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 acknowledge from the device (wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond to the I2C WRITE command with a “CONFIGURATION SETTING ERROR STATUS” message (F0h 7Dh 00h 25h 5Ah F7h).

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)

The USB-microDig, when in host mode, sends it's 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 (8^{th}) 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 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 that has been opened if different than 127 (7Fh).

The [BODY] of the reset message:

0xxxxxxx:	xxxxxxx = 0 (00h) to 6 (06h);	defines an opened port
	xxxxxxx = 127 (7Fh);	no port opened

Example:

240, 125, 0 {DEV}, 125 {STATUS}, 03 {DATA}, 247 (F0h, 7Dh, 00h, 7Fh, 03h, F7h)
This message indicates that port 3 (inputs 4-5) is opened for I2C communications.

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 command [BODY] is:

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}, 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 acknowledge from the device (wrong address or device not present) then the USB-microDig will respond with an I2C NO RESPONSE message (F0h, 00h, 7Bh, F7h). If the I2C port on the USB-microDig was not opened the USB-microDig will respond to the I2C WRITE command with a “CONFIGURATION SETTING ERROR STATUS” message (F0h 7Dh 00h 25h 5Ah F7h).

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:

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
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}, 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, 38h, 00h, 00h, 01h, 0Ah, 0Bh, 04h, 07h, F7h)
