



*Instruments That Advance The Art*

# microDXP

## RS-232 Communications Specification

Version 3.28

November 16, 2017

microDXP Hardware Revision: H (8)

**XIA LLC**

31057 Genstar Rd

Hayward, CA 94544 USA

Email: [support@xia.com](mailto:support@xia.com)

Tel: (510) 401-5760; Fax: (510) 401-5761

<http://www.xia.com/>

Information furnished by XIA LLC is believed to be accurate and reliable. However, no responsibility is assumed by XIA for its use, or for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of XIA. XIA reserves the right to change hardware or software specifications at any time without notice.

<b>COMMAND STRUCTURE.....</b>	<b>4</b>
<b>COMMANDS SUPPORTED.....</b>	<b>4</b>
<b>Run Control and Readout.....</b>	<b>4</b>
0x00: Start Run .....	4
0x01: End Run.....	4
0x02: Read MCA .....	4
0x04: Read MultiSCA.....	5
0x06: Read Run Statistics .....	5
0x07: Set/Get Run Preset .....	5
0x08: Take MCA Snapshot.....	6
0x09: Read Snapshot MCA.....	6
0x0A: Read Snapshot Run Statistics.....	6
0x0B: Set Run Mode.....	7
<b>Diagnostic Tools.....</b>	<b>8</b>
0x10: Read Diagnostic Histogram .....	8
0x11: Read Diagnostic Trace .....	9
0x12: Read Baseline History.....	10
<b>General Communications and Control .....</b>	<b>11</b>
0x40: I2C Read or Write .....	11
0x41: Read Temperature .....	11
0x42: Read DSP Parameter Names .....	11
0x43: Read/Write DSP Parameter.....	12
0x44: Read/Write DSP Program Memory .....	12
0x45: Read/Write DSP Data Memory.....	12
0x48: Read Serial Number .....	13
0x49: Get Board Information .....	13
0x4A: Echo.....	13
0x4B: Status .....	14
0x4C: Set/Get Input Enable .....	14
0x4D: Flash Write-Protect Control .....	14
0x4E: Reset FPGA .....	14
0x4F: Reset DSP .....	14
<b>Spectrometer Control.....</b>	<b>15</b>
0x80: Set/Get Digitizing Clock .....	15
0x81: Set/Get FiPPI Configuration .....	15
0x82: Set/Get Parameter Set .....	15
0x83: Set/Get General Set.....	15
0x84: Set/Get MCA Bin Width.....	16
0x85: Set/Get Number of MCA Bins.....	16
0x86: Set/Get Threshold .....	17
0x87: Set/Get Detector Polarity .....	17
0x88: Set/Get Base Gain .....	17
0x89: Set/Get RC Time Constant (Tau).....	18
0x8A: Set/Get Preamplifier Reset Time.....	18
0x8B: Set/Get Filter Parameter Value.....	18

0x8C: Read Parameter Set Values .....	19
0x8D: Save Parameter Set.....	19
0x8E: Read Current General Set Values .....	19
0x8F: Save Current General Set.....	19
0x90: Read SLOWLEN Values .....	20
0x91: Set/Get Gain Trim.....	20
0x92: Set/Get BLFILTER .....	21
0x93: Set/Get RUNTASKS.....	21
0x94: Set/Get FIPCONTROL .....	21
0x95: Set/Get AC-Coupling Controls .....	21
0x96: Set/Get SCA Pulse Periods .....	21
0x97: Set/Get Limits for Multiple SCA Regions .....	22
0x98: Set/Get Autostart.....	22
0x99: Set/Get SlopeDAC/OffsetDAC Value .....	22
0x9B: Set/Get Switched-Gain .....	22
0x9C: Set/Get Digital Base Gain .....	22
0x9D: Set/Get ADC Offset .....	23
0x9E: Get Reset/RC Switch Position .....	23
0x9F: Apply Settings.....	23

## Command Structure

The general structure for commands and responses is as follows:

[Esc][Command][Ndata (2 bytes)][data1]...[dataN][XOR CS]

### Where:

[Esc]	Escape (ASCII 0x1B) as a command start byte
[Command]	Single byte for command number, allowing up to 255 commands. See the tables below for command definitions.
[Ndata]	Number of data bytes to follow. Two bytes, low byte first.
[XOR CS]	Exclusive-or checksum (bitwise XOR of all bytes except for the initial [Esc]). If the checksum is not correct an error response is returned.

The format of responses echo the format of commands; that is, they start with the [Esc] character and pass back the command # to which it is responding, followed by appropriate data and checksum. The first data byte of all responses is the return status, which is zero for a successful command. In case of an error, only the error byte is returned – no other data bytes are sent. Please note that it is necessary to receive the entire response from one command before sending the next command, as there is no queuing of commands in the MicroDXP communications processor.

## Commands Supported

Blue font indicates variations for the new (blue) microDXP, aka SuperMicro

Orange font indicates modifications for certain customers

### Run Control and Readout

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x00: Start Run</b>	Enable data taking. Ndata = 1 [data1] = Clear MCA; 1: new run, 0: resume run	Returns the run number. Ndata = 3 [data1] = Return status (0: OK) [data2,data3] = Run number (low byte first)
<b>0x01: End Run</b>	Stop data taking. Ndata = 0	Ndata = 1 [data1] = Return status (0: OK) 0: OK, 1: error
<b>0x02: Read MCA</b>	Read MCA spectrum. Ndata = 5 [data1] = First MCA bin (low byte) [data2] = First MCA bin (high byte) [data3] = # of bins (low byte) [data4] = # of bins (high byte) [data5] = bytes per bin (1, 2 or 3)	MCA data, with the specified number of bytes per MCA bin. Ndata = 1 + (Number of bins)*(bytes per bin) [data1] = Return status (0: OK) [data2] = low byte of first MCA bin ... [dataN] = highest byte of highest MCA bin

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x04: Read MultiSCA</b>  <i>SuperMicro Only!!!</i>	Read number of events in defined SCA regions. Ndata = 0.	MultiSCA data; 4 bytes per SCA Ndata = 2 + 4*NUMSCA (max 66) [data1] = Return status (0: OK) [data2] = NUMSCA [data3] = low byte of SCA0 ... [dataN] = highest byte of last SCA
<b>0x06: Read Run Statistics</b>	Read real time, livetime, input events, and output events. Ndata = 0 or 1 Return data depends on Statistics readout mode: short readout mode omits the under- and overflows, while long readout mode includes everything. For PIC version 1.04 and later, an argument to this command is allowed: [data1]: Readout mode 0: short, 1: long. If omitted, the short readout mode is used.	Returns run statistics – Livetime, realtime, input events (fastpeaks), output events for short readout mode; long readout mode adds under- and overflows. Ndata = 21 for short mode, 29 for long mode. [data1] = Return status (0: OK) [data2 – data7] = Livetime, low byte first [data8 – data13] = Realtime (as above) [data14 – data17] = Fastpeaks [data18 – data21] = Output events For long readout mode only: [data22 – data25] = Underflows [data26 – data29] = Overflows (For DSP versions earlier than 1.08, long readout mode is not supported.)
<b>0x07: Set/Get Run Preset</b>	Set or read the preset run length. Default is no preset run length (indefinite run); values are persistent once set. Ndata = 6 or 8 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = Preset type: 0 = no preset (indefinite run), 1 = fixed realtime, 2 = fixed livetime, 3 = fixed output counts, 4 = fixed input counts [data3–4] = preset length low word, low byte first. [data5–6] = preset length mid word, low byte first. [data7–8] = preset length high word, low byte first (only necessary for large realtime or livetime presets, interpreted as zero if missing). Times are specified in 500 ns units.	Ndata = 6 or 8 [data1] = Return status (0: OK) [data2] = Preset type [data3–4] = preset length low word, low byte first. [data5–6] = preset length mid word, low byte first. [data7–8] = preset length high word, low byte first (only necessary for large realtime or livetime presets, interpreted as zero if missing).  Note that the response to a ‘set’ command is matched to the length of the command, for legacy support.  Note, however, that the response to a ‘get’ command is now always 8. If Handel uses the ‘get’ command, it should accept either 6 or 8 bytes in the response to support both older and newer DSP code.

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x08: Take MCA Snapshot</b> (Requires PIC version 1.04 and DSP version 1.08 or later)	Take snapshot of MCA contents. Allows readout of stable MCA spectrum during data acquisition. Only supported if MCA length is 4096 bins or less. Ndata = 0 or 1 [data1] = 0: no action, 1: clear spectrum and statistics after taking snapshot. If omitted, no action is taken.	Ndata = 1 [data1] = Return status (0: Ok, 1: MCALEN>4096 2: Timeout)
<b>0x09: Read Snapshot MCA</b> (Requires PIC version 1.04 and DSP version 1.08 or later)	Read snapshot of MCA spectrum. Ndata = 5 [data1] = First MCA bin (low byte) [data2] = First MCA bin (high byte) [data3] = # of bins (low byte) [data4] = # of bins (high byte) [data5] = bytes per bin (1, 2 or 3)	MCA data, with the specified number of bytes per MCA bin. Ndata = 1+ (Number of bins)*(bytes per bin) [data1] = Return status (0: OK, 1: MCALEN>4096) [data2] = low byte of first MCA bin ... [dataN] = highest byte of highest MCA bin
<b>0x0A: Read Snapshot Run Statistics</b> (Requires PIC version 1.04 and DSP version 1.08 or later)	Read real time, livetime, input events, and output events from snapshot of MCA data. Ndata = 0	Returns snapshot run statistics – Livetime, realtime, input events (fastpeaks), and output events. Ndata = 29 [data1] = Return status (0: OK, 1: MCALEN>4096) [data2 – data7] = Livetime, low byte first [data8 – data13] = Realtime (as above) [data14 – data17] = Fastpeaks [data18 – data21] = Output events [data22 – data25] = Underflows [data26 – data29] = Overflows

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x0B: Set Run Mode</b>	Change the data taking method. Ndata = 1 [data1] = Run Mode 0: normal 1: auto read out Updated 170414 (Custom PIC 35.3.14): Ndata = 1 or 6 DATA1 = RUNMODE (0 - normal, 1 - auto-readout/resume run, 3 - auto-readout/new run) DATA2 = First MCA bin (low byte) DATA3 = First MCA bin (high byte) DATA4 = # of bins (low byte) DATA5 = # of bins (high byte) DATA6 = # bytes per bin (1, 2 or 3)	Returns with specified run mode setting. Ndata = 2 [data1] = Return status (0: OK) [data2] = Run Mode
0x0C-0x0F: Reserved		

## Diagnostic Tools

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x10: Read Diagnostic Histogram</b>	<p><b>Old microDXP:</b> Read 1 kword histogram of baseline samples. Ndata = 0</p> <p><b>SuperMicro:</b> Read 1 kword histogram of diagnostic samples. Ndata = 0 or 1 [data1] = Data type 0 = Baseline 1 = ADC 2 = Fast-filter output 3 = Raw baseline 3 = Subtracted raw baseline 5 = Slow-filter baseline Note that changing the data type will reset the histogram. If data type is unmodified or omitted, the histogram is just read back.</p>	<p><b>Old microDXP:</b> Returns baseline histogram data – 1024 16-bit words.</p> <p><b>SuperMicro:</b> Returns diagnostic histogram data – 1024 16-bit words. Ndata = 2049 [data1] = Return status (0: OK) [data2] = first bin (low byte) [data3] = first bin (high byte) ... [data 2048] = last bin (low byte) [data 2049] = last bin (high byte)</p>



<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x11: Read Diagnostic Trace</b>	<p>Read diagnostic trace. Sampling time is specified in digitizing clock periods. The minimum period is dependent on the clock rate (setting sampling time to zero will give the minimum period).</p> <p><b>Old microDXP:</b> Always ADC trace with no trigger.  Ndata = 0 (keep settings) or 2  [data1] = Sampling interval (low byte)  [data2] = Sampling interval (high byte)</p> <p><b>SuperMicro:</b>  Ndata = 0 (keep settings), 2, 3 or 6  [data1] = Sampling interval (low byte)  [data2] = Sampling interval (high byte)  [data3] = Direct Readout via USB*  [data4] = Trigger Position (0-255)  0 → No pre-trigger data  128 → Trigger occurs at 50%  255 → All pre-trigger data  [data5] = Trigger Type  0 = No trigger (free run)  1 = Fast-filter event  2 = Intermediate or slow event  4 = Good event  8 = Overflow  16 = Underflow  32 = Fast pileup  64 = Slow pileup  128 = ADC out-of-range  [data6] = Trace Type  0 = ADC  1 = ADC Average  2 = Fast Filter  3 = Raw Intermediate Filter  4 = Baseline Samples  5 = Baseline Average  6 = Scaled Intermediate Filter  7 = Raw Slow Filter  8 = Scaled Slow Filter</p> <p>*If Ndata &gt;= 3 AND [data3] = 1, trace data is omitted from the RS-232 response.</p>	<p>The HISTORY buffer is filled with HSTLEN samples of the data, separated by the specified number of clock ticks.</p> <p><b>Old microDXP:</b>  Ndata = 1 + 2 * HSTLEN  [data1] = Return status (0: OK)  [data1] = first sample, low byte  [data2] = first sample, high byte  ...  [dataN] = last sample, high byte  Note: HSTLEN varies with firmware, but is typically 8000.</p> <p><b>SuperMicro:</b>  Same as old microDXP, unless the new Direct Readout via USB mode has been selected, then:  Ndata = 1  [data1] = Return status (0: OK)</p> <p>Note that the HISTORY buffer start address and length are stored in DSP parameters HSTSTART and HSTLEN, respectively.</p>

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x12: Read Baseline History</b> SuperMicro will return garbage – instead use command 0x11 with Trace Type = 3	Read a history of the most recent baseline average values. Ndata = 0.	The HISTORY buffer is filled with HSTLEN consecutive values of the baseline average. Ndata = 1 + 2 * HSTLEN [data1] = Return status (0: OK) [data2] = first point (low byte) [data3] = first point (high byte) ... [dataN] = last point, high byte
0x13-0x1F: Reserved		

## General Communications and Control

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x40: I2C Read or Write</b>	<p>Read from or write to an I2C device. Currently, only I2C devices with 7-bit addresses are supported. The maximum number of bytes to read is 64; the sum of the number of command bytes (e.g., target address for an EEPROM) and data bytes to write may not exceed 80.</p> <p>Ndata = 4 + NCMD (for a read) or 4 + NCMD + NI2CD (for a write)</p> <p>[data1] = 0: read, 1: write  [data2] = 7-bit I2C address (in bits 1-7; bit 0 indicates a read or a write, and is set internally)  [data3] = NCMD. The number of command bytes  [data4] = NI2CD = number of I2C data bytes to write/read  [data5 ...] = Command bytes then data bytes (for a write)</p>	<p>Ndata = 1 + number of bytes read (if any)  [data1] = Return status (0: OK)  [data2...] = return data</p>
<b>0x41: Read Temperature</b>	<p>Read the temperature measured by the onboard digital temperature sensor.</p> <p>Ndata = 0</p>	<p>Ndata = 3  [data1] = Return status (0: OK)  [data2] = signed integer temperature (2's complement) in degrees Celsius.  [data3] = fractional portion of temperature. Bit 7 corresponds to <math>2^{-1}</math>, bit 6 to <math>2^{-2}</math>, etc. Bits 0-3 always read 0 (temperature is reported to a precision of 1/16 degree). The fractional portion of the temperature is always positively added to the integer temperature.</p>
<b>0x42: Read DSP Parameter Names</b>	<p>Read the ordered list of DSP parameters names, which allows referencing the parameters by name.</p> <p>Ndata = 1  [data1] = Readout option  0: Return parameter name string and length parameters  1: Return only length parameters (allows pre-allocating memory for name string)</p>	<p>Ndata = 5 or 5 + StringLen depending on readout option. StringLen is the total number of characters (including null separators) used for the list of public DSP parameters.  [data1] = Return status (0: OK)  [data2, data3] = Number of parameters  [data4, data5] = StringLen  [data6...] List of DSP parameter names (in ASCII), null separated.</p>

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x43: Read/Write DSP Parameter</b>	<p>Read or write a single DSP parameter.</p> <p>Ndata = 2 (read) or 4 (write)</p> <p>[data1] = 0 (read) or 1 (write)</p> <p>[data2] = parameter number, defined by the position in the ordered parameter list (starting at 0)</p> <p>For write only:</p> <p>[data3,data4] = parameter value, low byte first</p>	<p>Ndata = 3</p> <p>[data1] = Return status (0: OK)</p> <p>[data2,data3] = parameter value, low byte first</p>
<b>0x44: Read/Write DSP Program Memory</b>	<p>Read directly from or write directly to DSP program memory. Program memory is organized in 24-bit (3 byte) words. Note the unusual byte ordering (dictated by the DSP host interface)</p> <p>Ndata = 4 (read) or 4+3*Nwords (write)</p> <p>[data1] = 0 (read) or 1 (write)</p> <p>[data2] = Nwords (max = 24 for 72 bytes)</p> <p>[data3, data4] = target address, relative to start of PM (low byte first)</p> <p>Write only:</p> <p>[data5...] Data to write (3 bytes per word, middle byte first, then high byte then low byte last)</p>	<p>Ndata = 1 (write) or 1+3*Nwords (read)</p> <p>[data1] = Return status (0: OK)</p> <p>Read only:</p> <p>[data2...] = Data read from PM (3 bytes per word, middle byte then high byte then low byte last)</p>
<b>0x45: Read/Write DSP Data Memory</b>	<p>Read directly from or write directly to DSP data memory. Data memory is organized in 16-bit (2 byte) words.</p> <p>Ndata = 4 (read) or 4 + 2*Nwords (write)</p> <p>[data1] = 0 (read) or 1 (write)</p> <p>[data2] = Nwords (max = 32 for 64 bytes)</p> <p>[data3, data4] = target address, relative to start of DM (low byte first)</p> <p>Write only:</p> <p>[data5...] Data to write (2 bytes per word, low byte first)</p>	<p>Ndata = 1 (write) or 1+2*Nwords (read)</p> <p>[data1] = Return status (0: OK)</p> <p>Read only:</p> <p>[data2...] = Data read from DM (2 bytes per word, low byte first)</p>

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x48: Read Serial Number</b>	Read serial number. Ndata = 0.	Ndata = 1 + SerialLength [data1] = Return status (0: OK) [data2 – dataN] = serial number, ASCII, null terminated (16 character maximum, including null)
<b>0x49: Get Board Information</b>	Read board information – versions, variants, clock information, FiPPI information. Ndata = 0	[Ndata] = 18 + 3*NFiPPI [data1] = Return status (0: OK) [data2] = PIC code variant [data3] = PIC code major version [data4] = PIC code minor version [data5] = DSP code variant [data6] = DSP code major version [data7] = DSP code minor version [data8] = DSP clock speed (MHz) [data9] = Clock Enable register [data10] = NFiPPI (number of FPGA configurations) <b>Old microDXP:</b> [data11] = Gain mode (0: fixed, 1: VGA) <b>SuperMicro:</b> [data11] = Gain mode (0: fixed, 1: VGA, 2: VGA + Digital; 3: Switched-Gain + Digital, 4: Ketek High/Low) [data12-13] = Nominal gain (gain with variable gain=1) 1.15 mantissa [data14] = Nominal gain exponent Nominal gain = (mantissa/32768)*2^(exponent) [data15] = Nyquist filter (0: 2 MHz, 1: 4 MHz, 2: > 4 MHz) [data16] = ADC speed grade (0: 20 MHz, 1: 40 MHz, 2: 65 MHz) [data17] = FPGA speed (0: normal, 1: fast) [data18] = Analog power supply (0: local regulators, 1: no local regulators) Then, for each FiPPI: [dataN] = FiPPI decimation [dataN+1] = FiPPI version [dataN+2] = FiPPI variant
<b>0x4A: Echo</b>	Echoes command	Ndata same as input command. Checksum is recalculated.

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x4B: Status</b>	Returns board status	Ndata = 6 [data1] = Return status [data2] = PIC status (0: OK) [data3] = DSP boot status (0: OK) [data4] = Run state (0: idle, 1: running) [data5] = DSP BUSY value [data6] = DSP RUNERROR value
<b>0x4C: Set/Get Input Enable</b>	Set or read MicroDXP input enable state (input signal can be disabled with CMOS switch). Ndata = 2 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = input enable (0: disable input, 1: enable input)	Ndata = 2 [data1] = Return status (0: OK) [data2] = Input Enable state (0: disabled, 1: enabled)
<b>0x4D: Flash Write-Protect Control</b>	The Flash memory write-protection feature prevents writing to the first 4 sectors. It is enabled by default. Ndata = 1 [data1] = write-protect mode 0: protected mode (default) 1: unprotected mode	Ndata = 2 [data1] = Return status (0: OK) [data2] = write-protect mode 0: protected mode (default) 1: unprotected mode
<b>0x4E: Reset FPGA</b>	Reset the FPGA (in preparation to reset the PIC for bootloader feature). Note that the DSP will immediately enter the error loop... Ndata = 4 [data1] = 0xAA [data2] = 0x55 [data3] = 0xAA [data4] = 0x55	Ndata = 1 [data1] = Return status (0: OK)
<b>0x4F: Reset DSP</b>	Reset the processor – the DSP is rebooted and the FPGA is reprogrammed. The four data bytes must match the values specified below to enable the reset. Ndata = 4 [data1] = 0xAA [data2] = 0x55 [data3] = 0xAA [data4] = 0x55	Ndata = 1 [data1] = Return status (0:OK)

## Spectrometer Control

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x80: Set/Get Digitizing Clock</b>	<p>Set or read current digitizing rate.</p> <p><b>Old microDXP:</b>            Ndata = 2 (set) or 1 (get)            [data1] = 0: set, 1: get            [data2] = Clock selection            0: DSPCLK, 1: DSPCLK/2, 2: DSPCLK/4, 3: DSPCLK/8</p> <p><b>SuperMicro: does not support changing the digitization rate</b>            Ndata = 0 (get)</p>	<p>Ndata = 2            [data1] = return status            0: OK, 1: invalid setting (selected clock speed not supported)            [data2] = Current clock setting.</p>
<b>0x81: Set/Get FiPPI Configuration</b>	<p>Select or read FPGA configuration. Up to three can be stored on the board.</p> <p>Ndata = 2 (set) or 1 (get)            [data1] = 0: set, 1: get            [data2] = FiPPI selection (0, 1 or 2)            Number of configurations depends on board options.</p>	<p>Ndata = 2            [data1] = return status            0: OK, 1: invalid setting            [data2] = Current FiPPI configuration (0, 1 or 2)</p>
<b>0x82: Set/Get Parameter Set</b>	<p><b>Old microDXP:</b> Select one of five peaking times available for a given Clock/FiPPI combination, or read current setting. The base peaking time is given by the selected clock and FiPPI: <math>\text{BaseTime} = 6 * \text{Clock Period} * 2^{\text{Decimation}}</math></p> <p>Within a FiPPI, five peaking times are available: 1, 1.5, 2, 3 and 4 times the base peaking time value, corresponding to parameter sets 0, 1, 2, 3 and 4.</p> <p><b>SuperMicro: Select one of twenty-four peaking times, or read current setting.</b></p> <p>Ndata = 2 (set) or 1 (get)            [data1] = 0: set, 1: get            [data2] = Selected parameter set</p>	<p>Ndata = 2            [data1] = return status            0: OK, 1: invalid setting            [data2] = Current parameter set</p>
<b>0x83: Set/Get General Set</b>	<p>Select one of five general settings, or read current setting. The General set includes parameters that do not depend on peaking time: gain, bin width, polarity, number of bins, preamplifier characteristics (Tau RC or reset time).</p> <p>Ndata = 2 (set) or 1 (get)            [data1] = 0: set, 1: get            [data2] = Selected general set</p>	<p>Ndata = 2            [data1] = return status            0: OK, 1: invalid setting            [data2] = Current parameter set</p>

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x84: Set/Get MCA Bin Width</b>	<p>Set the granularity of the MCA bins, or read the current setting.  Ndata = 3 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2] = Granularity  0: Very fine (e.g. 5 eV/bin)  1: Fine (e.g. 10 eV/bin)  2: Medium (e.g. 20 eV/bin)  3: Coarse (e.g. 40 eV/bin)  4: Custom  For the custom setting (otherwise ignored):  [data3] = Width in terms of the minimum (Very fine) bin width (e.g. a setting of 7 would give 35 eV/bin)</p>	<p>Ndata = 3  [data1] = return status  0: OK, 1: invalid setting  [data2] = Current bin granularity  [data3] = Custom bin scaling factor</p>
<b>0x85: Set/Get Number of MCA Bins</b>	<p>Set or read the number of MCA bins (max 8192) and the offset (normally 0).  Ndata = 5 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2, data3] = Number of MCA bins (low byte first)  [data4,data5] = Offset (the first bin of the spectrum; can be nonzero)</p>	<p>Ndata = 5  [data1] = return status  0: OK, 1: invalid setting  [data2, data3] = Current number of MCA bins (low byte first)  [data4,data5] = Current MCA offset</p>



<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x86: Set/Get Threshold</b>	<p>Set threshold to apply to one of the filters (up to three), or read the current settings.</p> <p><b>Old microDXP:</b>  Ndata = 3 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2]: Filter choice (0 = fast, 1 = intermediate, 2 = energy)  [data3]: Threshold value (up to 255)</p> <p>Set threshold to apply to one of the filters (up to three), or read the current settings.</p> <p><b>SuperMicro:</b>  All thresholds now range from 0 to 4095  Ndata = 4 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2]: Filter choice (0 = fast, 1 = intermediate, 2 = energy)  [data3] = Threshold value (low byte)  [data4] = Threshold value (high byte)</p>	<p><b>Old microDXP:</b>  Ndata = 4  [data1] = return status  0: OK, 1: invalid setting  [data2] = Current fast threshold  [data3] = Current intermediate threshold  [data4] = Current slow threshold</p> <p><b>SuperMicro:</b>  Ndata = 7  [data1] = return status  0: OK, 1: invalid setting  [data2] Fast threshold (low byte)  [data3] Fast threshold (high byte)  [data4] Intermediate threshold (low byte)  [data5] Intermediate threshold (high byte)  [data6] Slow threshold (low byte)  [data7] Slow threshold (high byte)</p>
<b>0x87: Set/Get Detector Polarity</b>	<p>Select detector preamplifier polarity or read current value.</p> <p>Ndata = 2 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2] : Polarity: 0 = negative-going steps, 1 = positive</p>	<p>Ndata = 2  [data1] = return status  0: OK, 1: invalid setting  [data2] = Current polarity setting</p>
<b>0x88: Set/Get Base Gain</b>	<p><b>Only used if GAINMODE equals 1 or 2:</b>  Set or read the 16-bit GAINBASE value. Changing this value modifies the gain for all choices of PARSET.</p> <p>Ndata = 3 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2] = GAINBASE (low byte)  [data3] = GAINBASE (high byte)</p>	<p>Ndata = 3  [data1] = return status  0: OK, 1: no GainDAC installed  [data2] = GAINBASE (low byte)  [data3] = GAINBASE (high byte)</p>

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x89: Set/Get RC Time Constant (Tau)</b>	Set or read Tau, the preamplifier RC decay time; applicable to RC feedback preamplifiers only. The time is specified in terms of the digitization clock period. For example, using a preamplifier with a 50 microsecond decay time at a digitization rate of 40 MHz, the decay time setting is $40 \times 50 = 2000$ . Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = low byte of decay time [data3] = high byte of decay time	Ndata = 3 [data1] = return status 0: OK, 1: invalid setting [data2] = low byte of Tau [data3] = high byte of Tau
<b>0x8A: Set/Get Preamplifier Reset Time</b>	Set or read the preamplifier reset time; applicable to reset type preamplifiers only. The time is specified in microseconds. Ndata = 2 (set) or 1 (get) [data1] = 0: set, 1: get [data2] reset time in microseconds	Ndata = 2 [data1] = return status 0: OK, 1: invalid setting [data2] = reset time in microseconds
<b>0x8B: Set/Get Filter Parameter Value</b>	Set or read the value of one of the parameters that controls the digital filtering. These parameters are part of the parameter set that is stored in flash memory, and only need to be changed for optimization purposes. <b>Old microDXP:</b> Ndata = 3 (set) or 2 (get) [data1] = 0: set, 1: get [data2] = Parameter number [data3] = Parameter value Parameter numbers are as follows: 0: SLOWLEN 1: SLOWGAP 2: PEAKINT 3: PEAKSAM 4: FASTLEN 5: FASTGAP 6: MINWIDTH 7: MAXWIDTH <b>SuperMicro:</b> Ndata = 4 (set) or 2 (get) [data1] = 0: set, 1: get [data2] = Parameter number [data3] = Parameter value (low byte) [data4] = Parameter value (high byte) Parameter numbers are as follows: 0-7: Same as before 8: BFACTOR $INTLEN = SLOWLEN / 2^{(BFACTOR+1)}$ 9: PEAKMODE 0 – finding 1 – sampling	<b>Old microDXP:</b> Ndata = 3 [data1] = return status 0: OK, 1: invalid parameter [data2] = Parameter number [data3] = Parameter value <b>SuperMicro:</b> Ndata = 4 [data1] = return status 0: OK, 1: invalid [data2] = Parameter number [data3] = Parameter value (low byte) [data4] = Parameter value (high byte)

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x8C: Read Parameter Set Values</b>	<p>Read values in a parameter set (PARSET). The parameter set starts with the parameter NUMPARSET and contains the NUMPARSET parameters immediately following NUMPARSET in DSP parameter memory. The names of the parameters can be obtained by reading the full list of DSP parameter names (command 0x42) and identifying the block starting with NUMPARSET.</p> <p>Ndata = 1 or 3  [data1] = 0: only return NUMPARSET, 1: return NUMPARSET and all parameter values from current set, 2: return all parameters from selected PARSET  [data2] = FiPPI number  [data3] = PARSET number  <b>Old microDXP:</b> (0 through 4)  <b>SuperMicro:</b> (0 through 23)</p>	<p>Ndata = 2 or 4 + 2*NUMPARSET  [data1] = Return status (0: OK)  [data2] = NUMPARSET  The following bytes are only returned when the full PARSET data are requested:  [data3 – data4] = PARSET version  [data5...] = 16-bit parameter values, low byte first</p>
<b>0x8D: Save Parameter Set</b>	<p>Save the current parameter set. The two tag bytes must be correct to enable this command.</p> <p>Ndata = 3  [data1] = PARSET number  <b>Old microDXP:</b> (0 through 4)  <b>SuperMicro:</b> (0 through 23)  [data2] = 0x55  [data3] = 0xAA</p>	<p>Ndata = 2  [data1] = Return status  [data2] = Parameter set number  <b>Old microDXP:</b> (0 through 4)  <b>SuperMicro:</b> (0 through 23)</p>
<b>0x8E: Read Current General Set Values</b>	<p>Read values in the currently selected general set. The general set starts with the parameter NUMGENSET and contains the NUMGENSET parameters immediately following NUMGENSET in DSP parameter memory. The names of the parameters can be obtained by reading the full list of DSP parameter names (command 0x42) and identifying the block starting with NUMGENSET.</p> <p>Ndata = 0 (only return NUMGENSET) or 1  [data1] = 0: only return NUMGENSET, 1: return NUMGENSET and all parameter values</p>	<p>Ndata = 2 or 4 + 2*NUMGENSET  [data1] = Return status (0: OK)  [data2] = NUMGENSET  The following bytes are only returned when the full GENSET data are requested:  [data3 – data4] = GENSET version  [data5...] = 16-bit parameter values, low byte first</p>
<b>0x8F: Save Current General Set</b>	<p>Save the current general set. The four tag bytes must be correct to enable this command.</p> <p>Ndata = 3  [data1] = General set number (0 through 4)  [data2] = 0x55  [data3] = 0xAA</p>	<p>Ndata = 2  [data1] = Return status  [data2] = General set number</p>

<i>Command</i>	<i>Meaning</i>	<i>Response data</i>
<b>0x90: Read SLOWLEN Values</b>	<p>Read out SLOWLEN values from all parameter sets (in order to build a list of available peaking times). The peaking time for a given PARSET can be calculated as:            Peaking Time = (1/DSPSPEED) * 2^(CLKSET+Decimation)*SLOWLEN            In microseconds, where</p> <ul style="list-style-type: none"> <li>- 1/DSPSPEED is the period of the DSP clock, in microseconds</li> <li>- The digitizing clock is obtained by dividing down by 2^CLKSET</li> <li>- 2^Decimation samples are combined into one entry into the digital filter</li> <li>- SLOWLEN entries are combined to form the trapezoidal filter</li> </ul> <p>Ndata = 0</p>	<p><b>Old microDXP:</b>            Ndata = 3 + 6*NFiPPI            [data1] = Return status            [data2] = CLKSET            [data3] = NFiPPI            Then for each FiPPI:            [dataN] = Decimation            [dataN+1] = SLOWLEN, PARSET 0            0            [dataN+2] = SLOWLEN, PARSET 1            1            [dataN+3] = SLOWLEN, PARSET 2            2            [dataN+4] = SLOWLEN, PARSET 3            3            [dataN+5] = SLOWLEN, PARSET 4            4</p> <p><b>SuperMicro:</b>            Ndata = 52            [data1] = Return status            [data2] = CLKSET            [data3] = NFiPPI==1            [data4] = Decimation            [data5] = SLOWLEN, PARSET 0 (low byte)            [data6] = SLOWLEN, PARSET 0 (high byte)            ...            [data51] = SLOWLEN, PARSET 23 (low byte)            [data52] = SLOWLEN, PARSET 23 (high byte)</p>
<b>0x91: Set/Get Gain Trim</b>	<p>Set or get GAINTWEAK value corresponding to current GENSET.  <a href="#">Note that GAINTWEAK calculation now depends on GAINMODE (see SuperMicro_Gain_Specification.docx)</a></p> <p>Ndata = 3 (set) or 1 (get)            [data1] = 0: set, 1: get            [data2] = GAINTWEAK (low byte)            [data3] = GAINTWEAK (high byte)</p>	<p>Ndata = 3            [data1] = Return status            [data2] = GAINTWEAK (low byte)            [data3] = GAINTWEAK (high byte)</p>

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x92: Set/Get BLFILTER</b>	Set or get BLFILTER value. Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = BLFILTER (low byte) [data3] = BLFILTER (high byte)	Ndata = 3 [data1] = Return status [data2] = BLFILTER (low byte) [data3] = BLFILTER (high byte)
<b>0x93: Set/Get RUNTASKS</b>	Set or get RUNTASKS. Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = RUNTASKS (low byte) [data3] = RUNTASKS (high byte)	Ndata = 3 [data1] = Return status [data2] = RUNTASKS (low byte) [data3] = RUNTASKS (high byte)
<b>0x94: Set/Get FIPCONTROL</b>	Set or get FIPCONTROL. Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = FIPCONTROL (low byte) [data3] = FIPCONTROL (high byte)	Ndata = 3 [data1] = Return status [data2] = FIPCONTROL (low byte) [data3] = FIPCONTROL (high byte)
<b>0x95: Set/Get AC-Coupling Controls</b>	Set or read TAUCTRL, which controls for the AC-coupled circuit variant. Ndata = 2 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = TAUCTRL[7:0] 0*: AC-coupled w/ 500ns decay time 1: AC-coupled w/ 1us decay time 2: AC-coupled w/ 4us decay time 3-254: AC-coupled w/ 10us decay time 255: DC-coupled  *Note that 0 forces ASC_RESET*=0, overriding RUNTASKS bits 11 and 12.	Ndata = 2 [data1] = return status 0: OK [data2] = TAUCTRL[7:0]
<b>0x96: Set/Get SCA Pulse Periods</b>	Set or get time periods for the SCA pulse output in 25ns units. Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = SCATIMEON (pulse assert time) <i>Note: Setting SCATIMEON=0 disables the AUXIO output buffers in the FiPPI!</i> [data3] = SCATIMEOFF (minimum wait time between pulses)	Ndata = 3 [data1] = Return status [data2] = SCATIMEON [data3] = SCATIMEOFF

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x97: Set/Get Limits for Multiple SCA Regions</b>  <i>SuperMicro</i> <i>CODEREV&lt;=0x0520</i> <i>Supports</i> <i>NUMSCA&lt;=4!!!</i>	Set or get limits for NUMSCA (up to 16) regions. Ndata = 2+4*NUMSCA (set) or 1 (get) [data1] = 0: set, 1: get [data2] = NUMSCA [data3] = SCA0LIMLO (low byte) [data4] = SCA0LIMLO (high byte) [data5] = SCA0LIMHI (low byte) [data6] = SCA0LIMHI (high byte) [data7] = SCA1LIMLO (low byte) etc.	Ndata = 2+4*NUMSCA (max 66) [data1] = Return status [data2] = NUMSCA [data3] = SCA0LIMLO (low byte) [data4] = SCA0LIMLO (high byte) [data5] = SCA0LIMHI (low byte) [data6] = SCA0LIMHI (high byte) [data7] = SCA1LIMLO (low byte) etc.
<b>0x98: Set/Get Autostart</b> (note: only valid for PIC version 1.3 or later)	Set or get Auto start setting (if set, a run is automatically started after initial boot). Ndata = 2 [data1] = 0: set, 1: get [data2] = Autostart (0: disabled, 1: enabled)	Ndata = 2 [data1] = Return status (0: OK) [data2] = Autostart setting
<b>0x99: Set/Get SlopeDAC/OffsetDAC Value</b>	Set or read the 16-bit SLOPEDAC (for reset preamplifier) or OFFSETDAC (for RC preamplifier) value. Ndata = 3 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = Slope/ Offset DAC (low byte) [data3] = Slope/ Offset DAC (high byte)	Ndata = 3 [data1] = return status 0: OK [data2] = Slope/ Offset DAC (low byte) [data3] = Slope/ Offset DAC (high byte)
<b>0x9B: Set/Get Switched-Gain</b>	Set or read the 4-bit SWGAIN, which determines the discrete switched-gain. Ndata = 2 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = SWGAIN[3:0]	Ndata = 2 [data1] = return status 0: OK [data2] = SWGAIN[3:0]
<b>0x9C: Set/Get Digital Base Gain</b>	Set or read the 16-bit DGAINBASE (0.16 mantissa) and signed 8-bit DGAINBASEEXP (exponent), which determine the Digital Base Gain. Ndata = 4 (set) or 1 (get) [data1] = 0: set, 1: get [data2] = DGAINBASE[7:0] [data3] = DGAINBASE[15:8] [data4] = DGAINBASEEXP[7:0]	Ndata = 4 [data1] = return status 0: OK [data2] = DGAINBASE[7:0] [data3] = DGAINBASE[15:8] [data4] = DGAINBASEEXP[3:0]

<b>Command</b>	<b>Meaning</b>	<b>Response data</b>
<b>0x9D: Set/Get ADC Offset</b>	<p><b>RC variants only:</b> Set or read the 14-bit ADC Offset value (to SETOFFADC). The DSP immediately adjusts the Offset DAC iteratively to match the baseline ADC value to the user setting. If ADC Offset = 0, DAC is just set to mid-range value (0x8000).  Ndata = 3 or 4 (set) or 1 (get)  [data1] = 0: set, 1: get  [data2] = ADC Offset (low byte)  [data3] = ADC Offset (high byte)  [data4] = enable auto-adjust  zero: DAC will remain static until next command 0x9A (or power cycle)  non-zero: DAC will be re-adjusted whenever ApplySettings routine is called, e.g. when analog gain or other settings are changed.  IF Ndata =4 GLOBSET IS SAVED!  If Ndata=3, current auto-adjust setting is maintained.</p>	<p>Ndata = 4  [data1] = return status  0: OK  [data2] = ADC Offset (low byte)  [data3] = ADC Offset (high byte)  [data4] = enable auto-adjust</p>
<b>0x9E: Get Reset/RC Switch Position</b>	<p>Set or read SWITCHPOS  Ndata = 0</p>	<p>Ndata = 2  [data1] = return status  0: OK  [data2] = Switch Position: Reset (0) or RC (1)</p>
<b>0x9F: Apply Settings</b>	<p>Apply settings defined by DSP parameter values.  Ndata = 0</p>	<p>Ndata = 1  [data1] = Return status</p>