

## MicroDXP RS232 Command Specification:

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.

Version: 1.2, March 4, 2008

### Commands Supported

#### Run Control and Readout

Code	Command	Meaning	Response data
0x00	<b>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)
0x01	<b>End Run</b>	Stop data taking. Ndata = 0	Ndata = 1 [data1] = Return status (0: ok) 0: ok, 1: error
0x02	<b>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 ... [lastdata] = highest byte of highest mca bin

Code	Command	Meaning	Response data
0x03	Read SCA <i>Not implemented.</i>	Read number of events in defined SCA. Ndata = 0.	Single SCA data; 4 bytes. Ndata = 5) [data1] = Return status (0: ok) [data2] = lowest byte of SCA counts ... [data3] = highest byte of SCA counts
0x04	Read MultiSCA <i>Not implemented.</i>	Read number of events in multiple SCA regions. Ndata = 0. Requires special firmware.	MultiSCA data; 4 bytes per SCA Ndata = 2 + 4*NUMSCA [data1] = Return status (0: ok) [data2] = NUMSCA [data3] = low byte of SCA0 ... [lastdata] = highest byte of last SCA
0x05	Read MultiSCAMAP <i>Not implemented.</i>	Read MultiSCA map. Each point includes statistics. Ndata = 0. Requires special firmware.	MultiSCA map data. Each point contains statistics and MultiSCA data. Ndata = 4+NUMPOINTS*(12+3*NUMSCA) [data1] = Return status (0: ok) [data2] = NUMSCA [data3] = NUMPOINTS, low byte [data4] = NUMPOINTS, high byte [data5 – lastdata] = MultiSCA data For each point, the order of bytes is: Livetime: low, middle, high byte Realtime: low, middle, high byte Fastpeaks: low, middle, high byte Events: low, middle high byte SCA0: low, middle, high byte ... SCAn: low, middle, high byte
0x06	<b>Read run statistics</b>	Read real time, livetime, input events, 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. See command 0x9A for setting the readout mode. For PIC version 1.04 and later, an argument to this command is allowed: [data1]: Readout mode 0: short, 1: long. If omitted, the readout mode set using command 0x9A 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.)

Code	Command	Meaning	Response data
0x07	<b>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 [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 – data6] = preset length, low byte first. Times are specified in 500 ns units.	Ndata = 6 [data1] = Return status (0: ok) [data2] = Preset type [data3 – data6] = preset length, low byte first.
0x08	<b>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	Ndata = 1 [data1] = Return status (0: Ok, 1: MCALEN>4096 2: Timeout)
0x09	<b>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 ... [lastdata] = highest byte of highest mca bin
0x0A	<b>Read snapshot run statistics</b> (Requires PIC version 1.04 and DSP version 1.08 or later)	Read real time, livetime, input events, output events from snapshot of MCA data. Ndata = 0	Returns snapshot run statistics – Livetime, realtime, input events (fastpeaks), 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
0x0B – 0x0F	Reserved		

## Diagnostic Tools

Code	Command	Meaning	Response data
0x10	<b>Read Baseline Histogram</b>	Read 1 kword histogram of baseline samples. Ndata = 0	Returns baseline 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)
0x11	<b>Read ADC Trace</b>	Read ADC 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). The maximum sampling interval is equal to 512 microseconds; longer times will be rounded down to this value. Ndata = 2. [data1] = Sampling time (low byte) [data2] = Sampling time (high byte)	The HISTORY buffer is filled with HSTLEN samples of the ADC, separated by the specified number of clock ticks. Ndata = 1 + 2 * HSTLEN [data1] = Return status (0: ok) [data1] = first adc sample, low byte [data2] = first adc sample, high byte ... [lastdata] = last adc sample, high byte Note: HSTLEN varies with firmware, but is typically 8000.
0x12	<b>Read baseline history</b>	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) ... [lastdata] = last point, high byte
0x13 - 0x1F	Reserved		

### General Communications and Control

Code	Command	Meaning	Response data
0x40	<b>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 (eg, 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</p> <p>[data2] = 7-bit I2C address (in bits 1-7; bit 0 indicates a read or a write, and is set internally)</p> <p>[data3] = NCMD. The number of command bytes</p> <p>[data4] = NI2CD = number of I2C data bytes to write/read</p> <p>[data5 ...] = Command bytes then data bytes (for a write)</p>	<p>Ndata = 1 + number of bytes read (if any)</p> <p>[data1] = Return status (0: ok)</p> <p>[data2...] = return data</p>
0x41	<b>Read Temperature</b>	<p>Read the temperature measured by the onboard digital temperature sensor.</p> <p>Ndata = 0</p>	<p>Ndata = 3</p> <p>[data1] = Return status (0: ok)</p> <p>[data2] = signed integer temperature (2's complement) in degrees Celsius.</p> <p>[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>

Code	Command	Meaning	Response data
0x42	<b>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</p> <p>[data1] = Readout option</p> <p>0: Return parameter name string and length parameters</p> <p>1: Return only length parameters (allows preallocating 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.</p> <p>[data1] = Return status (0: ok)</p> <p>[data2,data3] = Number of parameters</p> <p>[data4,data5] = StringLen</p> <p>[data6...] List of DSP parameter names (in ASCII), null separated.</p>
0x43	<b>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>
0x44	<b>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>

Code	Command	Meaning	Response data
0x45	<b>Read/Write DSP Data Memory</b>	Read directly from or write directly to DSP data memory. Data memory is organized in 16-bit (2 byte) words. Ndata = 4 (read) or 4 + 2*Nwords (write) [data1] = 0 (read) or 1 (write) [data2] = Nwords (max = 32 for 64 bytes) [data3,data4] = target address, relative to start of DM (low byte first) Write only: [data5...] Data to write (2 bytes per word, low byte first)	Ndata = 1 (write) or 1+2*Nwords (read) [data1] = Return status (0: ok) Read only: [data2...] = Data read from DM (2 bytes per word, low byte first)
0x46	Set/Get Idle Mode <i>Not implemented.</i>	Set or read idle mode. If automatic idle mode is enabled, the uDXP will enter a low power state aspecified time after the end of a run (if another run is not started). Ndata = 4 [data1] = 0: set, 1: get [data2] = idle mode 0: enable automatic powerdown 1: disable auto powerdown [data3,data4] = delay before entering powerdown mode, in seconds (low byte first)	Ndata = 4 [data1] = Return status (0: ok) [data2] = idle mode [data3,data4] = delay before entering low power mode
0x47	Set/Get Sleep mode <i>Not implemented.</i>	Set uDXP sleep state. Several different subsystems can be powered down – the analog section (if using onboard regulators), the ADC, the fpga and the DSP. Setting the bit corresponding to each subsection enables the power-down mode. Ndata = 2 [data1] = 0: set, 1: get [data2] = power-down state (1 means power-down) bit 0: Analog power-down bit 1: ADC power-down bit 2: FPGA power-down bits 3-4: DSP power-down (00 = full power, 01 = slow idle, 10 or 11 = full power-down)	Ndata = 2 [data1] = Return status (0: ok) [data2] = Power-down state

Code	Command	Meaning	Response data
0x48	<b>Read Serial Number</b>	Read serial number. Ndata = 0.	Ndata = 1 + SerialLength [data1] = Return status (0: ok) [data2 – lastdata] = serial number, ASCII, null terminated (16 character maximum, including null)
0x49	<b>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) [data11] = Gain mode (0: fixed, 1: variable) [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
0x4A	<b>Echo</b>	Echoes command	Ndata same as input command. Checksum is recalculated.
0x4B	<b>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

Code	Command	Meaning	Response data
0x4C	<b>Set/Get Input Enable</b>	Set or read MicroDXP input enable state (input signal can be disabled with CMOS switch). Ndata = 2 [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)
0x4D – 0x4E	Reserved		
0x4F	<b>Reset</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

Code	Command	Meaning	Response data
0x80	<b>Set/Get Digitizing clock</b>	Set or read current digitizing rate. Ndata = 2 [data1] = 0: set, 1: get [data2] = Clock selection 0: DSPCLK, 1: DSPCLK/2, 2: DSPCLK/4. 3: DSPCLK/8 The standard DSPCLK = 32 MHz. Available selections depend on board type.	Ndata = 2 [data1] = return status 0: OK, 1: invalid setting (selected clock speed not supported) [data2] = Current clock setting. 0: DSPCLK 1: DSPCLK/2 2: DSPCLK/4 3: DSPCLK/8
0x81	<b>Set/Get FiPPI configuration</b>	Select or read fpga configuration. Up to three can be stored on the board. Ndata = 2 [data1] = 0: set, 1: get [data2] = FiPPI selection (0, 1 or 2) Number of configurations depends on board options.	Ndata = 3 [data1] = return status 0: OK, 1: invalid setting [data2] = Current FiPPI configuration (0, 1 or 2) [data3] = Decimation value
0x82	<b>Set/get Parameter set</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: BaseTime = 6 * Clock Period * 2 <sup>Decimation</sup> 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. Ndata = 2 [data1] = 0: set, 1: get [data2] = Selected parameter set	Ndata = 2 [data1] = return status 0: OK, 1: invalid setting [data2] = Current parameter set
0x83	<b>Set/get General set</b>	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). Ndata = 2 [data1] = 0: set, 1: get [data2] = Selected general set	Ndata = 2 [data1] = return status 0: OK, 1: invalid setting [data2] = Current parameter set

Code	Command	Meaning	Response data
0x84	<b>Set/get MCA bin width</b>	Set the granularity of the MCA, or read the current setting. Ndata = 3 [data1] = 0: set, 1: get [data2] = Granularity 0: Very fine (eg 5 eV/ch). This is the minimum bin width. 1: Fine (eg 10 eV/ch) 2: Medium (eg 20 eV/ch) 3: Coarse (eg 40 eV/ch) 4: Custom For the custom setting: [data3] = Width in terms of the minimum (Very fine) bin width (eg a setting of 7 would give 35 eV/ch).	Ndata = 3 [data1] = return status 0: OK, 1: invalid setting [data2] = Current bin granularity [data3] = Custom bin scaling factor
0x85	<b>Set/get Number of MCA bins</b>	Set or read the number of MCA bins (max 8192) and the offset (normally 0). Ndata = 5 [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)	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
0x86	<b>Set/get threshold</b>	Set threshold to apply to one of the filters (up to three), or read the current settings. Ndata = 3 [data1] = 0: set, 1: get [data2] : Filter choice (0 = fast, 1 = intermediate, 2 = energy) [data3] : Threshold value (up to 255)	Ndata = 4 [data1] = return status 0: OK, 1: invalid setting [data2] = Current fast threshold [data3] = Current intermediate threshold [data4] = Current slow threshold
0x87	<b>Set polarity</b>	Select detector preamplifier polarity or read current value. Ndata = 2 [data1] = 0: set, 1: get [data2] : Polarity: 0 = negative-going steps, 1 = positive	Ndata = 2 [data1] = return status 0: OK, 1: invalid setting [data2] = Current polarity setting

Code	Command	Meaning	Response data
0x88	<b>Set/get Base GAINDAC value</b>	Set or read the 16-bit GAINBASE value. Changing this value modifies the gain for all choices of PARSET. The gain can also be tweaked individually for each PARSET; see command XXX below. The overall gain is $GAINDAC = GAINBASE + GAINTWEAK$ . Ndata = 3 [data1] = 0: set, 1: get [data2] = low byte of GAINBASE value [data3] = high byte Requires variable gain option.	Ndata = 3 [data1] = return status 0: OK, 1: no Gain DAC installed [data2] = low byte of Gain DAC value [data3] = high byte of Gain DAC value
0x89	<b>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 8 MHz, the decay time setting is $8 \times 50 = 400$ . Ndata = 3 [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
0x8A	<b>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 [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

Code	Command	Meaning	Response data
0x8B	<b>Set/get filter parameter value</b>	<p>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.</p> <p>Ndata = 3  [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  (more filter parameters may be added in the future)</p>	<p>Ndata = 3  [data1] = return status  0: OK, 1: invalid parameter  [data2] = Parameter number  [data3] = Parameter value</p>
0x8C	<b>Read current parameter set values</b>	<p>Read values in the currently selected parameter set. 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</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>
0x8D	<b>Save current 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] = Parameter set number (0 through 4)  [data2] = 0x55  [data3] = 0xAA</p>	<p>Ndata = 2  [data1] = Return status  [data2] = Parameter set number</p>

Code	Command	Meaning	Response data
0x8E	<b>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 = 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</p> <p>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>
0x8F	<b>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>
0x90	<b>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:  PeakingTime = <math>(1/32) * 2^{(CLKSET+Decimation)} * SLOWLEN</math>  In microseconds, where</p> <ul style="list-style-type: none"> <li>- 1/32 is the period of the 32 MHz clock</li> <li>- The digitizing clock is obtained by dividing down by <math>2^{CLKSET}</math></li> <li>- <math>2^{Decimation}</math> 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>3 + 6*NFiPPI  [data1] = Return status  [data2] = CLKSET  [data3] = NFiPPI</p> <p>Then for each FiPPI:  [dataN] = Decimation  [dataN+1] = SLOWLEN, parset 0  [dataN+2] = SLOWLEN, parset 1  [dataN+3] = SLOWLEN, parset 2  [dataN+4] = SLOWLEN, parset 3  [dataN+5] = SLOWLEN, parset 4</p>

Code	Command	Meaning	Response data
0x91	<b>Set or get GAINWEAK Value</b>	Set or get GAINWEAK value corresponding to current GENSET. GAINWEAK is added to GAINBASE to get the final GAINDAC setting, and is used to adjust for scaling variations between peaking time settings. NData = 3 [data1] = 0: set, 1: get [data2] = low byte of new GAINWEAK value [data3] = high byte	Ndata = 3 [data1] = Return status [data2] = GAINWEAK, low byte [data3] = high byte
0x92	<b>Set or get BLFILTER</b>	Set or get BLFILTER value. NData = 3 [data1] = 0: set, 1: get [data2] = low byte of new BLFILTER value [data3] = high byte	Ndata = 3 [data1] = Return status [data2] = BLFILTER, low byte [data3] = high byte
0x93	<b>Set or get RUNTASKS</b>	Set or get RUNTASKS. NData = 3 [data1] = 0: set, 1: get [data2] = low byte of new RUNTASKS value [data3] = high byte	Ndata = 3 [data1] = Return status [data2] = RUNTASKS, low byte [data3] = high byte
0x94	<b>Set or get FIPCONTROL</b>	Set or get FIPCONTROL. NData = 3 [data1] = 0: set, 1: get [data2] = low byte of new FIPCONTROL value [data3] = high byte	Ndata = 3 [data1] = Return status [data2] = FIPCONTROL, low byte [data3] = high byte
0x95	<b>Set or get TRACEWAIT</b>	Set or get TRACEWAIT. NData = 3 [data1] = 0: set, 1: get [data2] = low byte of new TRACEWAIT value [data3] = high byte	Ndata = 3 [data1] = Return status [data2] = TRACEWAIT, low byte [data3] = high byte
0x96	Set or get limits for Single SCA region <i>Not implemented.</i>		
0x97	Set or get limits for Multiple SCA regions <i>Not implemented.</i>		

Code	Command	Meaning	Response data
0x98	<b>Set or 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
0x99	<b>Set/get SlopeDAC/OffsetDAC value</b>	Set or read the 16-bit SLOPEDAC (for reset preamplifier) or OFFSETDAC (for RC preamplifier) value. Ndata = 3 [data1] = 0: set, 1: get [data2] = low byte of SLOPEDAC/OFFSETDAC value [data3] = high byte	Ndata = 3 [data1] = return status 0: OK [data2] = low byte of DAC value [data3] = high byte of DAC value
0x9A	<b>Set/get Statistics readout mode</b>	Set or read the statistics readout mode (see command 0x6). A value of 0 specifies short readout mode (under- and overflows are not included); a nonzero value specifies long readout mode. Ndata = 2 [data1] = 0: set, 1: get [data2] = Statistics readout mode: 0: short, 1: long	Ndata = 2 [data1] = return status 0: OK [data2] = Statistics readout mode 0: short, 1: long
0x9B - 0x9E	Reserved		
0x9F	<b>Apply Settings</b>	Apply settings defined by DSP parameter values. Ndata = 0	Ndata = 1 [data1] = Return status