Thorlabs Motion Controllers

Host-Controller Communications Protocol

Date: 14 May 2025

Contents

Messages Applicable to BPC20x Series

Messages Applicable to BPC30x Series

Messages Applicable to PPC001 and PPC102

Messages Applicable to TPZ001 and KPZ101

Messages Applicable to KPZ101 Only

Messages Applicable to KPC101

Messages Applicable to TSG001 and KSG101

Messages Applicable to KSG101 Only

Messages Applicable to MPZ601

Messages Applicable to TDC001 and KDC101

Messages Applicable to TSC001 and KSC101

Messages Applicable to KSC101 Only

Messages Applicable to TST001, TST101, KST101 and K10CR1

Messages Applicable to TST101 and KST101

Messages Applicable to TST001

Messages Applicable to KST101 Only

Messages Applicable to KST201 Only

Messages Applicable to K10CR1 Only

Messages Applicable to BSC10x and BSC20x

Messages Applicable to LTS150 and LTS300

Messages Applicable to MLJ050, MLJ150 and MLJ250

Messages Applicable to MFF101 and MFF102

Messages Applicable to BBD10x, BBD20x, BBD30x, TBD001 and KBD101

Messages Applicable to BBD10x, BBD20x, and TBD001 (Continued)

Messages Applicable to KBD101 Only

Messages Applicable to PDXC2

PZMOT_PARAM Sub-Messages Applicable to TIM101

Set/Request/Get_PZMOT_ CloseLoopParams (sub-message ID = 39)
Set/Request/Get_PZMOT_ Etmconfig (sub-message ID = 3B)
Set/Request/Get_PZMOT_ EtmParams (sub-message ID = 3C)
Set/Request/Get_PZMOT_ TPOS (sub-message ID = 3D)
Set/Request/Get_PZMOT_ EthrenetParams (sub-message ID = 3F)
Set/Request/Get_PZMOT_ CURRENT_POS (sub-message ID = 40)
Set/Request/Get_PZMOT_ AMD (sub-message ID = 43)
Set/Request/Get_PZMOT_JogParams (sub-message ID = 44)
Set/Request/Get_PZMOT_AmpOutParams (sub-message ID = 45)
Set/Request/Get_PZMOT_OpenMoveParams (sub-message ID = 46)
Set/Request/Get_PZMOT_CloseMoveParams (sub-message ID = 47)

Messages Applicable to BBD301, BBD302 and BBD303 Only

Messages Applicable to BNT001, MNA601, TNA001 and KNA101

Messages Applicable to KNA101 Only

Messages Applicable to TLS001 and KLSxxx

Messages Applicable Only to KLS635 and KLS1550

Messages Applicable to TLD001 and KLD101

Messages Applicable Only to KLD101

Messages Applicable to TQD001, TPA101 and KPA101

QUAD_PARAM Sub-Messages

Messages Applicable to TPA101 and KPA101 Only

QUAD_PARAM Sub-Messages

Messages Applicable to KPA101 Only

QUAD_PARAM Sub-Messages

Messages Applicable to TTC001

TEC_PARAM Sub-Messages

Messages Applicable to TIM101 and KIM101

PZMOT_PARAM Sub-Messages Applicable to TIM101

PZMOT_PARAM Sub-Messages Applicable to KIM101

Messages Applicable to MPC220 and MPC320

Messages Applicable to CT1P

Introduction

1. Purpose and Scope

This document describes the low-level communications protocol and commands used between the host PC and controller units within the Thorlabs Motion Control family. The information contained in this document is intended to help third party system developers to write their own applications to interface to the Thorlabs range of controllers without the constraints of using a particular operating system or hardware platform. The commands described here are those which are necessary to control movement; there is an additional set of commands, used for calibration or test, which will not be detailed as these are not required for the external system developer.

2. Electrical interface

The Thorlabs family of controllers provides a USB and an RS-232 interface to communicate with the host PC. The communications protocol is identical in both cases but developers wishing to use the USB interface should be aware of the USB enumeration scheme used in the system.

2.1 USB Interface

The electrical interface within the Thorlabs controllers uses a Future Technology Devices International (FTDI), type FT232BM USB peripheral chip to communicate with the host PC. This is a USB2.0 compliant USB1.1 device. This USB interfacing chip provides a serial port interface to the embedded system (i.e., Thorlabs controller) and USB interface to the host control PC. While the overall communications protocol is independent of the transport layer (for example, Ethernet or serial communications could also be used to carry commands from the host to the controller), the initial enumeration scheme described below is specific to the USB environment.

FTDI supply device drivers and interfacing libraries (for Windows, Linux, and other platforms) used to access the USB chip. Before any PC USB communication can be established with an Thorlabs controller, the client program is required to set up the necessary FTDI chip serial port settings used to communicate to the Thorlabs controller embedded system. Within the Thorlabs software itself the following FTDI library calls are made to set up the USB chip serial port for each Thorlabs USB device enumerated on the bus.

```
// Set baud rate to 115200.
ftStatus = FT_SetBaudRate(m_hFTDevice, (ULONG)uBaudRate);

// 8 data bits, 1 stop bit, no parity
ftStatus = FT_SetDataCharacteristics(m_hFTDevice, FT_BITS_8, FT_STOP_BITS_1,
FT_PARITY_NONE);

// Pre purge dwell 50ms.
Sleep(uPrePurgeDwell);

// Purge the device.
ftStatus = FT_Purge(m_hFTDevice, FT_PURGE_RX | FT_PURGE_TX);

// Post purge dwell 50ms.
Sleep(uPostPurgeDwell);
```

```
// Reset device.
ftStatus = FT_ResetDevice(m_hFTDevice);

// Set flow control to RTS/CTS.
ftStatus = FT_SetFlowControl(m_hFTDevice, FT_FLOW_RTS_CTS, 0, 0);

// Set RTS.
ftStatus = FT_SetRts(m_hFTDevice);
```

2.2 USB Device Enumeration

The Thorlabs Server PC software supplied is designed to work with a number of different types of controllers. The purpose of the enumeration phase is for the host to establish what devices are present in the system and initialise the GUI accordingly. Initially this is done by enumerating the USB devices connected to the system and reading the serial number information contained in the USB device descriptor.

For the Thorlabs range of controllers, this serial number is an 8-digit decimal number. The first two digits (referred to as the prefix) describe the type of controller, while the rest of the digits make up a unique serial number. By extracting the prefix, the host can therefore establish what type of hardware is connected to the system.

In most cases, specifically with benchtop controllers, the USB serial number contains sufficient information for the host to know the exact type of hardware is connected. There is a range of other controller products where several controller cards (without their own individual USB peripheral chip) can be plugged into a motherboard and it is only the motherboard that has USB connectivity. These are generally referred to as a card slot (or bay) type of system (for example, the BSC103 controller). In these systems, a second enumeration state is carried out; however, this second state is done within the protocol framework that will be detailed in this document.

The USB prefixes for some of our controllers are given below. For details on the prefix for a specific controller, please see the associated product handbook available from our website or contact your local tech support.

USB S/N	Type of product	Thorlabs code
20xxxxxx	Legacy single channel benchtop stepper driver	BSC001
21xxxxxx	Legacy single channel benchtop piezo driver	BPC001
22xxxxxx	Benchtop NanoTrak	BNT001
25xxxxxx	Legacy single channel mini stepper driver	BMS001
26xxxxxx	K-Cube stepper driver	KST101
27xxxxxx	K-Cube brushed DC servo driver	KDCT101
28xxxxxx	K-Cube brushless DC servo driver	KBD101
29xxxxxx	K-Cube piezo driver	KPZ101
30xxxxxx	Legacy dual channel stepper driver	BSC002
31xxxxxx	Legacy dual channel benchtop piezo driver	BPC002
33xxxxxx	Single channel benchtop DC servo driver to 2006	BDC101
35xxxxxx	Legacy dual channel mini stepper driver	BMS002
37xxxxxx	Motorized filter flipper	MFF10X
40xxxxxx	Single channel stepper driver	BSC101
41xxxxxx	Single channel piezo driver	BPC101

43xxxxxx	Single channel benchtop DC servo driver from 2007	BDC101
44xxxxxx	Single channel precision piezo driver	PPC001
45xxxxxx	LTS series integrated long travel stepper stages	LTS150/LTS300
48xxxxxx	MMR series Midi Rack bay serial number prefix	
49xxxxxx	Integrated stepper driven labjack	MLJ050/MLJ150/MLJ250
50xxxxxx	Midi Rack stepper module	MST601/MST602
51xxxxxx	Midi Rack piezo module	MPZ601
52xxxxxx	Midi Rack NanoTrak module	MNA601/IR
54xxxxxx	Thorlabs APT Midi Rack Brushless Module	MBD602
55xxxxxx	Integrated stepper driven rotation stage	K10CR1
56xxxxxx	K-Cube Laser Source	KLS101
57xxxxxx	K-Cube NanoTrak	KNA101
59xxxxxx	K-Cube Strain Gauge Reader	KSG101
60xxxxxx	OptoSTDriver (mini stepper driver)	OST001
63xxxxxx	OptoDCDriver (mini DC servo driver)	ODC001
64xxxxxx	T-Cube Laser Driver	TLD001
65xxxxxx	T-Cube Inertial Piezo Driver	TIM001
67xxxxxx	T-Cube brushless DC servo Driver	TBD001
68xxxxxx	K-Cube solenoid Driver	KSC101
69xxxxxx	K-Cube position aligner	KPA101
70xxxxxx	Three channel card slot stepper driver	BSC103/BSC203
71xxxxxx	Three channel card slot piezo driver	BPC103/203/303
72xxxxxx	Three channel card slot piezo/stepper driver	BPS103
73xxxxxx	Three channel card slot brushless DC driver	BBD103
80xxxxxx	Stepper Driver T-Cube	TST001
81xxxxxx	Piezo Driver T-Cube	TPZ001
82xxxxxx	NanoTrak T-Cube	TNA001
83xxxxxx	DC Driver T-Cube	TDC001
84xxxxxx	Strain Gauge Reader T-Cube	TSG001
85xxxxxx	Solenoid Driver T-Cube	TSC001
86xxxxxx	T-Cube Laser Source	TLS001
87xxxxxx	T-Cube TEC driver	TTC001
89xxxxxx	T-Cube Quad Detector	TQD001
90xxxxxx	Single channel stepper motor driver card	SCC101
91xxxxxx	Single channel piezo driver card	PCC101
93xxxxxx	Single channel DC servo driver card	DCC101
94xxxxxx	Brushless DC motor card	BCC101
95xxxxxx	2-Channel precision piezo controller	PPC102
96xxxxxx	2-Channel Precision piezo controller card	PCC102
97xxxxxx	Thorlabs Kinesis K-Cube Inertial Motor Controller	KIM101
98xxxxxx	Thorlabs Kinesis K-Cube Laser Diode Controller	KLD101
101xxxxx	Thorlabs Slim Integrated XY Motor Stage	M30XY
102xxxxx	Thorlabs Air Quality Monitor	AQM001
103xxxxx	Thorlabs Benchtop Brushless DC Servo Motor Controller	BBD30X
104xxxxx	Thorlabs Benchtop Brushless DC Servo Motor Virtual Controller Card	BBD30X
105xxxxx	Thorlabs Slim Integrated X Motor Stage	M30X
106xxxxx	Thorlabs Slim Integrated Motor Stage Virtual Controller Card	M30XY
107xxxxx	Thorlabs BBD Ethernet Controller Card	ETH001
108xxxxx	Thorlabs BBD Front Panel Controller Card	FP001

109xxxxx	Thorlabs Kinesis K-Cube Voice Coil Controller	KVC101
110xxxxx	Thorlabs Motorized Science Desk	SDAM7590 SDPM7590
111xxxxx	Thorlabs Manual Linear Encoder	ENC
112xxxxx	Thorlabs Piezo Stage Controller	PDXC2
113xxxxx	Thorlabs K-Cube Piezo Controller and Strain Gauge Reader	KPC101
114xxxxx	Thorlabs 2-Axis Joystick	MJC002

2.3 RS-232 Interface

The RS-232 interface uses the 9-way D-Type male connector on the rear panel, marked 'INTERCONNECT'. Communications parameters are fixed at:

- 115200 bits/sec
- 8 data bits, 1 stop bit
- No parity
- RTS/CTS Handshake

By nature, the RS-232 interface provides point-to-point communications, and therefore there is no device enumeration as there is with USB based communications.

3. Overview of the Communications Protocol

The communications protocol used in the Thorlabs controllers is based on the message structure that always starts with a fixed length, 6-byte *message header* which, in some cases, is followed by a variable length *data packet*. For simple commands, the 6-byte message header is sufficient to convey the entire command. For more complex commands, for example, when a set of parameters needs to be passed on, the 6-byte header is not enough and in this case the header is followed by the data packet.

The header part of the message always contains information that indicates whether a data packet follows the header and if so, the number of bytes that the data packet contains. In this way the receiving process can keep tracks of the beginning and the end of messages.

Note that in the section below describing the various byte sequences, the C-type of notation will be used for hexadecimal values (e.g., 0x55 means 55 hexadecimal) and logical operators (e.g., | means logic bitwise OR). Values that are longer than a byte follow the Intel little-endian format.

4. Description of the message header

The 6 bytes in the message header are shown below:

Byte:	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
Meaning if no data packet to follow	message ID		param1	param2	dest	source
Meaning if data packet to follow	message ID		data packet length		dest | 0x80	source

The meaning of some of the fields depends on whether the message is followed by a data packet or not. This is indicated by the most significant bit in byte 4, called the destination byte*,* therefore the receiving process must first check if the MSB of byte 4 is set.

If this bit is not set, then the message is a header-only message, and the interpretation of the bytes is as follows:

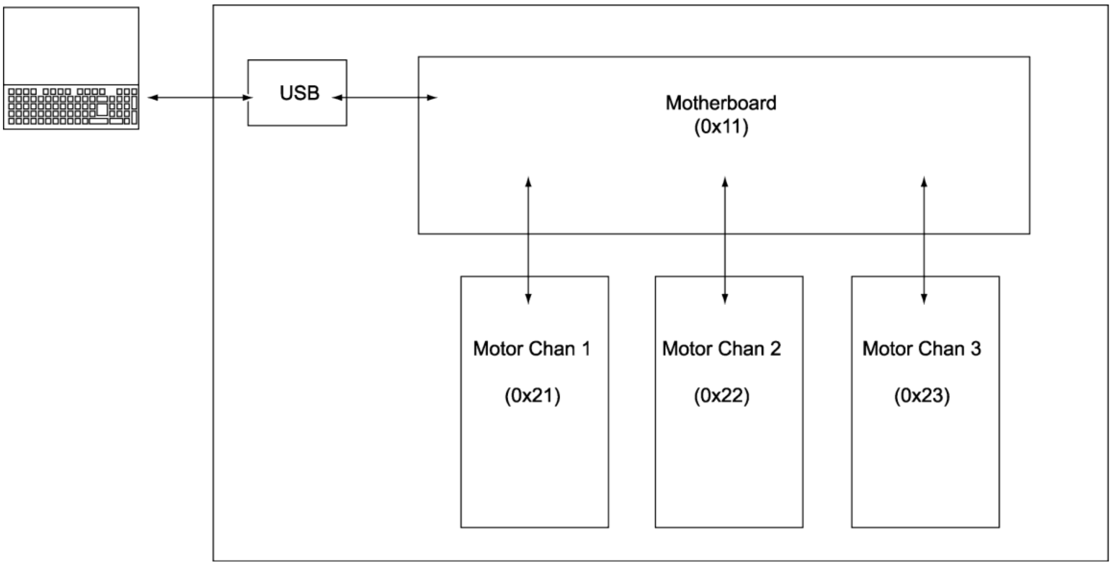
message ID:	describes what the action the message requests.
param1:	first parameter (if the command requires a parameter, otherwise 0).
param2:	second parameter (if the command requires a parameter, otherwise 0).
dest:	the destination module.
source:	the source of the message.

The meaning of the source and destination bytes will be detailed later.
If the MSB of byte 4 is set, then the message will be followed by a data packet and the interpretation of the header is the following:

message ID:	describes what the action the message requests.
datapacket length:	number of bytes to follow header. Note: although this is a 2-byte long field, currently no datapacket exceeds 255 bytes in length.
dest: | 0x80	the destination module logic OR'd with 0x80 (noted by d|)
source:	the source of the data.

The source and destination fields require some further explanation. In general, as the name suggests, they are used to indicate the source and destination of the message. In non-card-slot type of systems the source and destination of messages is always unambiguous, as each module appears as a separate USB node in the system. In these systems, when the host sends a message to the module, it uses the source identification byte of 0x01 (meaning host) and the destination byte of 0x50 (meaning "generic USB unit"). (In messages that the module sends back to the host, the content of the source and destination bytes is swapped.)

In card-slot (bay) type of systems, there is only one USB node for several sub-modules, so this simple scheme cannot be used. Instead, the host sends a message to the motherboard that the sub-modules are plugged into, with the destination field of each message indicating which *slot* the message must be routed to. Likewise, when the host receives a message from a particular sub-module, it knows from the source byte which slot is the origin of the message – see Fig below.



Numerically, the following values are currently used for the source and destination bytes:

```
0x01        Host controller (i.e., control PC)
0x11        Rack controller, motherboard in a card slot system or
            comms router board
0x21        Bay 0 in a card slot system
0x22        Bay 1 in a card slot system
0x23        etc.
0x24        etc.
0x25        etc.
0x26        etc.
…
0x2A        Bay 9 in a card slot system
0x50        Generic USB hardware unit
```

In slot-type systems the host can also send messages to the motherboard that the sub-modules are plugged into (destination byte = 0x11). In fact, as a very first step in the communications process, the host must send a message to the motherboard to find out which slots are used in the system.

Note that although in theory this scheme would allow communication between individual sub-modules (the source of the message could be a sub-module and the destination another one), current systems do not use this option.

5. General message exchange rules

The type of messages used in the communications exchange between the host and the sub-modules can be divided into 4 general categories:

(a) Host issues a command, sub-module carries out the command without acknowledgement (i.e., no response is sent back to the host).

Typically, these are commands which require no information from the sub-module, for example setting the digital outputs to a particular state.

(b) Host issues a command (message request) and the sub-module responds by sending data back to the host.

For example, the host may request the sub-module to report the state of the digital inputs.

(c) Following a command from the host, the sub-module periodically sends a message to the host without further prompting.

These messages are referred to as *status update messages*. These are typically sent automatically every 100 msec from the sub-module to the host, showing, amongst other things, the position of the stage the controller is connected to. The meters on the Thorlabs User GUI rely on these messages to show the up-to-date status of the stage.

(d) Rarely – error messages, exceptions. These are spontaneously issued by the sub-module if some error occurs. For example, if the power supply fails in the sub-module, a message is sent to the host PC to inform the user.

Apart from the last two categories (status update messages and error messages), in general the message exchanges follow the SET -> REQUEST -> GET pattern, i.e., for most commands a trio of messages are defined. The SET part of the trio is used by the host (or, sometimes in card-slot systems the motherboard) to set some parameter or other. If then the host requires some information from the sub-module, then it may send a REQUEST for this information, and the sub-module responds with the GET part of the command. Obviously, there are cases when this general scheme does not apply, and some part of this message trio is not defined. For consistency, in the description of the messages this SET->REQUEST->GET scheme will be used throughout.

Note that, as the scheme suggests, this is a master-slave type of system, so sub-modules never send SET and REQUEST messages to the host and GET messages are always sent to the host as a destination.

In all messages, where a parameter is longer than a single character, the bytes are encoded in the Intel format, least significant byte first.

6. Format Specifiers

format	encoding
word	Unsigned 16- bit integer (2 bytes) in the Intel (little-endian) format for example, decimal 12345 (3039H) is encoded as the byte sequence 39, 30
short	Signed 16-bit integer (2 bytes) in 2's compliment format for example, decimal -1 is encoded as the byte sequence FF, FF
dword	Unsigned 32-bit integer (4 bytes) in the Intel (little-endian) format for example, decimal 123456789 (75BCD15H) is encoded as the byte sequence 15, CD, 5B, 07
long	Signed 32-bit integer (4 bytes) in 2's compliment format for example, decimal -1 is encoded as the byte sequence FF, FF 4 bytes in the Intel (little-endian) format for example, decimal -123456789 (FFFFFFFFF8A432EBH) is encoded as the byte sequence EB, 32, A4, F8,
char	1 byte (2 digits)
char[N]	string of N characters

7. Single Precision Floating Point Format

Single-precision floating-point format is a computer number format that occupies 4 bytes (32 bits) in computer memory and represents a wide dynamic range of values by using a floating point.
Where message parameters use floating point variables, the system uses the IEEE 754 standard.

8. Conversion between position, velocity and acceleration values in standard physical units and their equivalent Thorlabs Software parameters.

To convert between the position and encoder counters in the stage being driven, and real-world units, (e.g. mm) the system uses certain conversion (scaling) factors. These conversion factors differ depending on the stage being driven and the controller being used.

Background

The principle described below is the same for all Thorlabs motion stepper and brushed or brushless DC controllers and stages, but the individual distance and time conversion factors will be typically different for each stage and/or controller.
In real life, the physical units needed to describe position, velocity and acceleration are related to position and time measurement units (millimetres/degrees and seconds). In motion controllers, however, normally the system only knows the distance travelled in encoder counts (pulses) as measured by an encoder fitted to the motor shaft. In most cases the motor shaft rotation is also scaled down further by a gearbox and a leadscrew. In any case, the result is a scaling factor between encoder counts and position. The value of this scaling factor depends on the stage. In the section below this scaling factor will be represented by the symbol EncCnt.
Time is related to the sampling interval of the system, and as a result, it depends on the motion controller. Therefore, this value is the same for all stages driven by a particular controller. In the sections below the sampling interval will be denoted by T.
The sections below describe the position, velocity, and acceleration scaling factors for all the controllers and stages that are used with these controllers. The symbols POS_{APT}, VEL_{APT} and ACC_{APT} are used to denote the position, velocity and acceleration values used in Thorlabs commands, whereas the symbols Pos, Vel and Acc denote physical position, velocity and acceleration values in mm, mm/sec and mm/sec^2 units for linear stages and degree, degree/sec and degree/sec^2 for rotational stages.

As Thorlabs parameters are integer values, the Thorlabs values calculated from the equations need to be rounded to the nearest integer.

Brushed DC Controller (TDC001, KDC101, KVS30) driven stages.

Mathematically:
POS_{APT} = EncCnt × Pos
VEL_{APT} = EncCnt × T × 65536 × Vel
ACC_{APT} = EncCnt × T^2 × 65536 × Acc
where T = 2048 / (6 × $10^{6)}$

The value of EncCnt and the resulting conversion factors are listed below for each stage:

Stage	EncCnt per mm or EncCnt per °	Scaling Factor	
		Velocity	Acceleration
MTS25-Z8	34554.96	772981.3692 (mm/s)	263.8443072 (mm/s^2)
MTS50-Z8	34554.96	772981.3692 (mm/s)	263.8443072 (mm/s^2)
Z8xx	34554.96	772981.3692 (mm/s)	263.8443072 (mm/s^2)
Z6xx	24600	550292.68 (mm/s)	187.83 (mm/s^2)
PRM1-Z8	1919.6418578623391	42941.66 (°/s)	14.66 (°/s^2)
PRMTZ8	1919.6418578623391	42941.66 (°/s)	14.66 (°/s^2)
CR1-Z7	12288	36650.0	95.276
KVS30	20,000	447392.43 (mm/s)	152.71 (mm/s^2)

Brushless DC Controller (TBD001, KBD101, BBD10X and BBD20X) driven stages.

Mathematically:
POS_{APT} = EncCnt × Pos
VEL_{APT} = EncCnt × T × 65536 × Vel
ACC_{APT} = EncCnt × T^2 × 65536 × Acc
where T = 102.4 × 10^{-6}

Linear Stages
The value of EncCnt and the resulting conversion factors are listed below for each stage:

Stage	EncCnt per mm	Scaling Factor	
		Velocity (mm/s)	Acceleration (mm/s^2)
DDSM50	2000	13421.77	1.374
DDSM100	2000	13421.77	1.374
DDS220	20000	134217.73	13.744
DDS300	20000	134217.73	13.744
DDS600	20000	134217.73	13.744
MLS203	20000	134217.73	13.744

Rotary Stages
The value of EncCnt and the resulting conversion factors are listed below for each stage:

Stage	EncCnt per 360°	Scaling Factor		
		EncCnt per °	Velocity (°/s)	Acceleration (°/s^2)
DDR100	3276800	9102.22	61083.98	6.255
DDR05	2000000	5555.55	37282.7	3.81775
DDR25	1440000	4000	26843.5	2.74878

Stepper Motor Controller (TST001, BSC00x, BSC10x, and MST601) Driven Stages

For these stepper controllers the server sends absolute micro-steps to the controllers. Depending on the stage and the stepper motor concerned there are different micro step values required to move either a linear distance in millimetres or a rotational distance in degrees.
In general, for 200 full step motors (most of our motors) the above range of stepper controllers is designed to insert 128 micro steps for every full step of the stepper.
So, for a 200 full step motor the number of micro steps per full turn is defined as follows:

Full turn micro steps = Motor full steps per turn x Number of Micro steps per full step

For a 200 full step motor this is given by: Full turn micro steps = 200 x 128 = 25600

However, the ZST and ZFS range of actuators have 24 full steps per revolution and furthermore, both motors are fitted with a gearbox. The ZST has a ratio 40.866:1, while the ZFS has a ratio 400:9.
So, for the ZST series, a 1mm move requires 24 x 128 x 40.866 = 125540.35 µsteps, while for the ZFS series, a 1mm move requires 24 x 128 x 400/9 = 136533.33 µsteps.

Each stage can either be a direct drive or driven through a gear box. The table below indicates the relationship between absolute micro steps and a positional output in millimetres or degrees.

This table is relevant for the range of controllers listed above. Note that micro step values are for a position of 1mm, a velocity of 1mm/sec and an acceleration of 1mm/sec/sec

Stage	Gearing	Position	Micro Step Values		
			Position(µs)	Velocity(µs/sec)	Acceleration(µs/sec^2)
ZST Series	0.0245 mm/turn	1mm	125540.35	125540.35	125540.35
ZFS Series	0.0225 mm/turn	1 mm	136533.33	136533.33	136533.33
DRV001	0.5mm/turn	1mm	51200	51200	51200
DRV013	1mm/turn	1mm	25600	25600	25600
DRV014	1mm/turn	1mm	25600	25600	25600
NRT100	1mm/turn	1mm	25600	25600	25600
NRT150	1mm/turn	1mm	25600	25600	25600
LTS150	1mm/turn	1mm	25600	25600	25600
LTS300	1mm/turn	1mm	25600	25600	25600
DRV113	1.25mm/turn	1mm	20480	20480	20480
DRV114	1.25mm/turn	1mm	20480	20480	20480
FW103*	No gear	0.998deg	71	71	71
NR360**	5.4546deg/turn	0.999deg	4693	4693	4693

*Note that there is no exact value of micro steps to get to exactly 1 degree this is because 1 turn represents 360 degrees which is 25600 micro steps. So actual resolution is 360/25600 = 0.0140625 degrees per micro step.

**Note that there is no exact value of micro steps to get to exactly 1 degree this is because 1 turn represents 5.4546 degrees which is 25600 micro steps. So actual resolution is 5.4546/25600 = 0.0002131 degrees

Stepper Motor Controller (TST101, KST101, BSC20x, MST602, K10CR1) Driven Stages

The latest stepper controllers include a Trinamics encoder with a resolution of 2048 microsteps per full step, giving 409600 micro-steps per revolution for a 200 step-motor. However, the ZST and ZFS range of actuators have 24 full steps per revolution and furthermore, both motors are fitted with a gearbox. The ZST has a ratio 40.866:1, while the ZFS has a ratio 400:9.
So, for the ZST series, a 1mm move requires 24 x 2048 x 40.866 = 2008645.63 µsteps, while for the ZFS series, a 1mm move requires 24 x 2048 x 400/9 = 2184533.33 µsteps.

This table is relevant only for the Trinamic-based range of controllers listed above. Note that micro step values are for a position of 1mm, a velocity of 1mm/sec and an acceleration of 1mm/sec/sec.

Stage	Gearing	Position	Trinamic converted Values		
			Position(µs)	Velocity(µs/sec)	Acceleration(µs/sec^2)
ZST Series	0.0245 mm/turn	1mm	2008645.63	107824097.5	22097.3
ZFS Series	0.0225 mm/turn	1mm	2184533.33	117265749.2	24111.85
DRV001	0.5mm/turn	1mm	819200	43974656	9012
DRV208	0.5mm/turn	1mm	819200	43974656	9012
DRV013	1mm/turn	1mm	409600	21987328	4506
DRV014	1mm/turn	1mm	409600	21987328	4506
NRT100	1mm/turn	1mm	409600	21987328	4506
NRT150	1mm/turn	1mm	409600	21987328	4506
LTS150	1mm/turn	1mm	409600	21987328	4506
LTS300	1mm/turn	1mm	409600	21987328	4506
MLJ050	1mm/turn	1mm	409600	21987328	4506
MLJ150	1mm/turn	1mm	409600	21987328	4506
MLJ250	1mm/turn	1mm	409600	21987328	4506
DRV113	1.25mm/turn	1mm	327680	17589862	3605
DRV114	1.25mm/turn	1mm	327680	17589862	3605
FW103*	No gear	1.0002deg	1138	61088	13
NR360	5.4546deg/turn	0.99997deg	75091	4030885	826
HDR50	5.4546deg/turn	0.99997deg	75091	4030885	826
K10CR1	120:1 (3deg/turn)	1 deg	136533	7329109	1502

In the above table the numbers that need to be sent to the controllers are based upon the Trinamics chip set conversions. The position is just the absolute number of micro-steps as before, as compared with the BSC10X range, the only difference is the 16 times greater resolution. However, for velocity and acceleration different conversion factors are required to get to correct motion profiles. For example, if a velocity of 409600 micro-steps per sec is required, then multiply by 53.68 i.e., 409600*53.68 gives 21987328 which for a 1mm lead screw would give 1mm/sec.
To accelerate at a rate of 409600 micro-steps/sec/sec (1mm/sec/sec), divide 409600 by 90.9 which gives 4506.

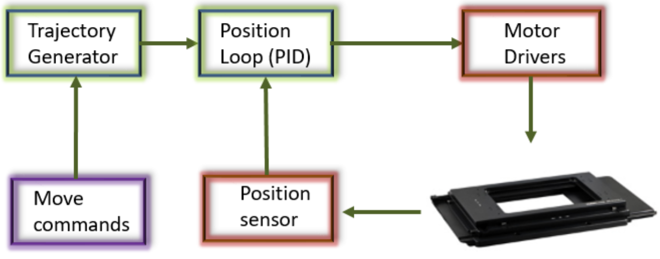
9. Initialising the MLJ050, MLJ150 and MLJ250 Motorised Labjack

In order for the Labjack to respond with end of moves or home completed messages, the user must first send a set of valid home parameters (MGMSG_MOT_SET_HOMEPARAMS 0x0440), for example Tx 40,04,0E,00,D0,01,01,00,02,00,01,00,F4,70,EE,03,00,C0,03,00

This message should be sent as part of the initialisation process, and acts as a flag to the rest of the code to indicate that a server is connected. Failure to do this will result in the end of move or home completed messages not being received.

AN INTRODUCTION TO MULTI-AXIS SYNCHRONIZED MOVES

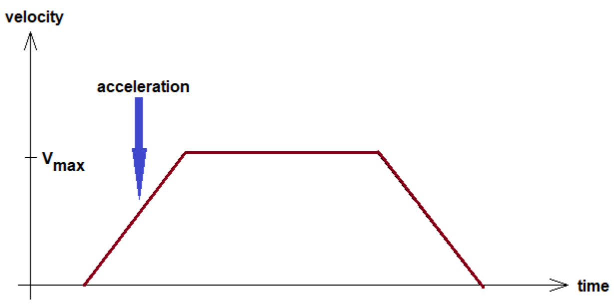
This section describes the implementation of multi-axis synchronized moves on the Thorlabs BBD30x series controllers.
To give developers a better understanding of the underlying principles of multi-axis synchronized moves, the relevant building blocks of the BMC10X controller are shown below:



At the heart of nearly all motion control algorithms a trajectory generator is used to generate the position points where the motor is required to be at any one time. The trajectory (position target) values output by the trajectory generator are fed to the position loop controller which compares the target position to the actual position and adjusts the motor current, always trying to maintain the target position. Thus, when a move is commanded, the corresponding parameters are sent to the trajectory generator, which then calculates the position target values required to execute the move.

Trapezoidal move profiles

Simple linear point-to-point moves (which are also the most used moves) are conveniently described by the end position, acceleration and maximum velocity values.



At the start of the move, the motor accelerates to the specified maximum velocity, travels at that velocity, and then decelerates to zero velocity and reaches that at exactly the target position.
For simple multi-axis moves, the trapezoidal move scheme can easily be extended to two or more dimensions.
If we want to move from position (x1, y1) to position (x2, y2) with acceleration *Acc* and velocity *Vel*, then the moves for the individual axes will be effectively the vector projections of the overall 2-D move.

Thus, from the 2-D move parameters we can calculate the move parameters for each axis by simply multiplying *Acc* and *Vel* with the scaling factors:

$$Scale_x = \frac{x_2 - x_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

and

$$Scale_y = \frac{y_2 - y_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

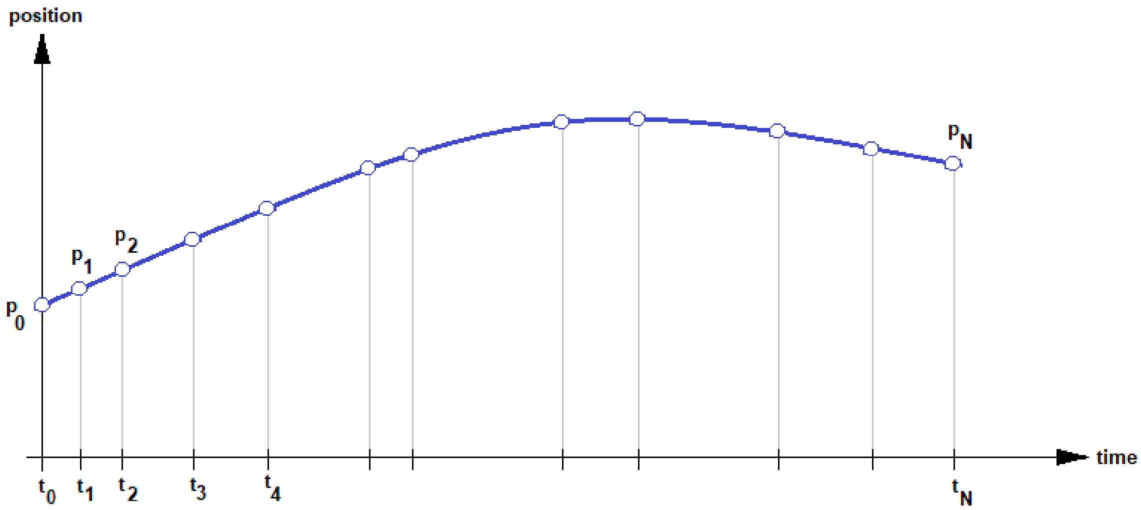
Thus:

$Acc_x = Acc * Scale_x$ and $Acc_y = Acc * Scale_y$

$Vel_x = Vel * Scale_x$ *and* $Vel_y = Vel * Scale_y$

Complex move shapes

The trapezoidal move profile is impractical for describing more complex move shapes, such as an arc or circle. For these, a different approach is used, and the move is described as a time-position array that defines the position targets the trajectory has to output at the predefined time points. For the time points in-between the specified points the trajectory generator uses linear interpolation.



The figure above shows the approach: the shape of the curve is described as a time-position array

$(t_0, p_0), (t_1, p_1) \ldots (t_N, p_N)$

The interval between the time points does not need to be equal. In fact, as linear interpolation is used between adjacent points, the algorithm effectively moves in a straight line between points, so more linear sections of the shape do not need to be described with the same frequency as more curved sections.
With this definition of the move trajectory the acceleration and velocity values are no longer predefined parameters but instead are implicit in the time-position difference between adjacent points. This also means that the user must pay attention to the velocity and acceleration limitations of the controller and the stage. With simple trapezoidal move profiles, the acceleration and velocity are normally set to values that are supported by the

controller and with reasonable values they result in smooth motion. The time-position array, on the other hand, allows the user more freedom but as a result also opens the possibility of move definitions that the controller cannot execute. For example, a large position change in a very short time can translate into excessive velocity and/or acceleration values.
Whilst the approach can be used to define single axis moves, it is more useful for multi-axis synchronized moves. For these the time-position array contains the position points for each axis. Thus, the array defines the position of all axes involved in the move at the same time points. Assuming 2 dimensional (x, y) coordinates, the array becomes:

$(t_0, x_0, y_0), (t_1, x_1, y_1) \dots (t_N, x_N, y_N)$

Obviously, the scheme can be extended to any number of axes that the controller supports.

Repeated patterns

To make the scheme more flexible, a section of the curve can also be repeated for several times. With this extension, the array can be considered as having a leading, a repeated and a tailing section. This is useful for applications where, for example, the repeated pattern needs to be preceded by an acceleration phase and then completed by deceleration to standstill.
In the illustration below the section p_K to p_L is repeated.

Leading section Tailing Section

---> -------------------------->

p_0 p_1 p_2 .... p_K p_{K+1} p_{K+2} p_L p_{L+1} p_{L+2} ... p_M

↑ ↓

<------------------------------------

Repeated section

Starting a synchronized trajectory

The user must consider the initial position of the stage when the trajectory is started. In almost all usage scenarios the stage will be at standstill when the synchronous move is started and immediately afterwards there will be a move to the first point in the time-position array. This can result in a large jump. The easiest way of avoiding this is by moving the stage to the first point defined in the time-position array prior to starting the synchronized trajectory.

To define the multi-axis synchronized moves, the time-position array and the corresponding parameters must be downloaded to the controller. This is supported by the following commands:

MGMSG_MOT_SET_MOVESYNCHARRAY	0x0A00
MGMSG_MOT_SET_MOVESYNCHPARAMS	0x0A03
MGMSG_MOT_MOVE_SYNCHSTART	0x0A06

Generic System Control Messages

Introduction

The messages described here are either system control messages, or else generic messages which apply to several or all controller types. Please see the list of controller specific commands for details on applicability to a specific controller type.

MGMSG_MOD_IDENTIFY 0x0223

Function: Instruct hardware unit to identify itself (by flashing its front panel LEDs).
In card-slot (bay) type of systems (which are usually the multi-channel controllers such as BSC102, BSC103, BPC302, BPC303, PPC102) the front panel LED that flashes in response to this command is controlled by the motherboard, not the individual channel cards. For these controllers the destination byte of the MGMSG_MOD_IDENTIFY message must be the motherboard (0x11) and the Channel Ident byte is used to select the channel to be identified. In single-channel controllers the Channel Ident byte is ignored as the destination of the command is uniquely identified by the USB serial number of the controller.

Channel Idents
0x01 channel 1
0x02 channel 2

Command structure (6 bytes):

0	1	2	3	4	5
header only					
23	02	Chan Ident	*00*	*d*	*s*

Example:
Identify controller #1 (channel 1 of the BSC103 controller) by flashing its front panel LED.

TX 23, 02, 01, 00, 11, 01

Identify the TDC001 controller (possibly within a group of various Thorlabs controllers in system):

TX 23, 02, 00, 00, 50, 01

MGMSG_MOD_SET_CHANENABLESTATE 0x0210
MGMSG_MOD_REQ_CHANENABLESTATE 0x0211
MGMSG_MOD_GET_CHANENABLESTATE 0x0212

Function Sent to enable or disable the specified drive channel.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
10	02	Chan Ident	Enable State	d	s

Channel Idents
0x01 channel 1
0x02 channel 2

For the TIM101 4 channel controller, the following idents are also used
0x04 channel 3
0x08 channel 4

Enable States
0x01 enable channel
0x02 disable channel

For single channel controllers such as the BBD10X, TDC001, the Chan Ident byte is always set to CHAN1.

Note: Although the BBD102 is in fact a 2-channel controller, 'channel' in this sense means "motor output channel within this module". Electrically, the BBD102 is a bay system, with two bays, each of them being a single channel controller, so only one channel can be addressed. There are controllers in the Thorlabs product range which indeed have multiple output channels (for example the MST601 module) for which the channel ident is used to address a particular channel.

Example: Enable the motor channel in bay 2

TX 10, 02, 01, 01, 22, 01

Note: MGMSG_MOT_REQ_USTATUSUPDATE 0x0490 command should be used for MPC220 and MPC320 instead of MGMSG_MOD_REQ_CHANENABLESTATE 0x0211 and 0x0212.

Example: If Channel 4 needs to be enabled and Channels 1 and 2 need to be disabled, the below procedure must be followed.

TX 10, 02, 04, 02, 50, 01
TX 10, 02, 03, 01, 50, 01

This shows MPC status with paddle #3 disabled and paddles #1 and #2 enabled.

TX 90, 04, 01, 00, 50, 01
TX 90, 04, 02, 00, 50, 01
TX 90, 04, 04, 00, 50, 01
RX 91, 04, 0E, 00, 81, 50, 01, 00, AD, 02, 00, 00, 01, 00, 00, 00, 00,04,00, 80
RX 91, 04, 0E, 00, 81, 50, 02, 00, AD, 02, 00, 00, 01, 00, 00, 00, 00,04,00, 80
RX 91, 04, 0E, 00, 81, 50, 04, 00, AD, 02, 00, 00, 01, 00, 00, 00, 00,04,00, 00

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	02	Chan Ident	0	d	s

As above, for single channel controllers such as the BBD10X, TDC001, the Chan Ident byte is always set to CHAN1.

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
12	02	Chan Ident	Enable State	d	s

The meaning of the parameter bytes “Chan Ident” and “Enable State” is the same as for the SET version of the commands.

MGMSG_HW_DISCONNECT 0x0002

Function: Sent by the hardware unit or host to disconnect from the Ethernet/USB bus.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
02	00	00	00	d	s

Example: Disconnect the BBD103 from the USB bus

TX 02, 00, 00, 00, 11, 00

MGMSG_HW_RESPONSE 0x0080

Function: Sent by the controllers to notify Thorlabs Server of some event that requires user intervention, usually some fault or error condition that needs to be handled before normal operation can resume. The message transmits the fault code as a numerical value – see the Return Codes listed in the Thorlabs Server helpfile for details on the specific return codes.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
80	00	00	00	d	s

Example: The BBD103 unit has encountered an over current condition

TX 80, 00, 00, 00, 01, 11

MGMSG_HW_RICHRESPONSE 0x0081

Function: Similarly, to HW_RESPONSE, this message is sent by the controllers to notify Thorlabs Server of some event that requires user intervention, usually some fault or error condition that needs to be handled before normal operation can resume. However, unlike HW_RESPONSE, this message also transmits a printable text string. Upon receiving the message, Thorlabs Server displays both the numerical value and the text information, which is useful in finding the cause of the problem.

REQ:

Response structure (74 bytes):
6-byte header followed by 68-byte (0x44) data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
header						*data*									
81	00	44	00	d|	s	MsgIdent		Code		<--------Notes------->					

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
data															
<--------------------------------Notes-------------------------------->															

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
data															
<-------------------------------- Notes -------------------------------->															

48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
data															
<-------------------------------- Notes -------------------------------->															

64	65	66	67	68	69	70	71	72	73
data									
<---------------------------Notes-------------------------->									

Data structure:

field	description	format
MsgIdent	If the message is sent in response to an Thorlabs Software message, these bytes show the message number that evoked the message. Most often though the message is transmitted due to some unexpected fault condition, in which case these bytes are 0x00, 0x00	word
Code	This is an internal Thorlabs specific code that specifies the condition that has caused the message (see Return Codes).	word]
Notes	This is a zero-terminated printable (ascii) text string that contains the textual information about the condition that has occurred. For example: “Hardware Time Out Error”.	char[64 bytes]

MGMSG_HW_START_UPDATEMSGS 0x0011

Function: Sent to start automatic status updates from the embedded controller. Status update messages contain information about the position and status of the controller (for example limit switch status, motion indication, etc). The messages will be sent by the controller every 100 msec until it receives a STOP STATUS UPDATE MESSAGES command. In applications where spontaneous messages (i.e., messages which are not received as a response to a specific command) must be avoided the same information can also be obtained by using the relevant GET_STATUTSUPDATES function.

Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	00	Unused	Unused	d	s

REQUEST: **N/A**

MGMSG_HW_STOP_UPDATEMSGS 0x0012

Function: Sent to stop automatic status updates from the controller – usually called by a client application when it is shutting down, to instruct the controller to turn off status updates to prevent USB buffer overflows on the PC.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
12	00	00	00	d	s

REQUEST: **N/A**
GET: **N/A**

MGMSG_HW_REQ_INFO 0x0005
MGMSG_HW_GET_INFO 0x0006

Function: Sent to request hardware information from the controller.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
05	00	00	00	d	s

Example: Request hardware info from controller #1

TX 05, 00, 00, 00, 11, 01

GET:
Response structure (90 bytes):
6 byte header followed by 84 byte (0x54) data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
header						*data*									
06	00	54	00	d|	s	<-Serial Number >				<--------Model Number------->					

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
data															
<Model> No		<Type>		<Firmware> Version >				<--------------------For internal use only-------------------->							

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
data															
<----------- For internal use only ----------->															

48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
data															
<----------- For internal use only ----------->															

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
data															
<----------- For internal use only ----------->															

80	81	82	83	84	85	86	87	88	89
data									
< For internal use only -->				HW Version		Mod State		<-nchs-->	

Data structure:

field	description	format
serial number	unique 8-digit serial number	long
model number	alphanumeric model number	char[8]
type	hardware type: 45 = multi-channel controller motherboard 44 = brushless DC controller	word
firmware version	firmware version byte[20] = minor revision number byte[21] = interim revision number byte[22] = major revision number byte[23] = unused	byte[4]
HW Version	The hardware version number	word
Mod State	The modification state of the hardware	word
nchs	number of channels	word

Example: Returned hardware info from controller #1

RX 06, 00, 54, 00, 81, 22, 89, 53, 9A, 05, 49, 4F, 4E, 30, 30, 31, 20, 00, 2C, 00, 02, 01, 39, 00, ……………………………., 00, 01, 00, 00, 00, 01, 00

Header: *06, 00, 54, 00, 81, 22*: Get Info, 54H (84) byte data packet, Motor Channel 2.
Serial Number: 89, 53, 9A, 05: 94000009
Model Number: 49, 4F, 4E, 30, 30, 31, 20, 00: ION001
Type*: 2C, 00:* 44 – Brushless DC Controller Card
firmware Version: 02, 01, 39, 00: 3735810
HW Version: 01, 00 Hardware version 01
Mod State: 03, 00, Modification stage 03.
No Chan: 01, 00: 1 active channel

MGMSG_RACK_REQ_BAYUSED 0x0060
MGMSG_RACK_GET_BAYUSED 0x0061

Function: Sent to determine whether the specified bay in the controller is occupied.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
60	00	Bay Ident	00	d	s

Bay Idents
0x00 Bay 1
0x01 Bay 2 to
0x09 Bay 10

Example: Is controller bay #1 (i.e., bay 0) occupied

TX 60, 00, 00, 00, 11, 01

GET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
61	00	Bay Ident	Bay State	d	s

Bay Idents
0x01 Bay 1
0x02 Bay 2 to
0x09 Bay 10

Bay States
0x01 Bay Occupied
0x02 Bay Empty (Unused)

Example: Controller Bay #1 (i.e. bay 0) is occupied

RX 61, 00, 00, 01, 11, 01

MGMSG_HUB_REQ_BAYUSED 0x0065
MGMSG_HUB_GET_BAYUSED 0x0066

Function: Sent to determine to which bay a specific unit is fitted.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
65	00	00	00	d	s

TX 65, 00, 00, 00, 50, 01

GET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
66	00	Bay Ident	00	d	s

Bay Idents
-0x01 T-Cube being standalone, i.e., off the hub.
0x00 T-Cube on hub, but bay unknown
0x01 Bay 1
0x02 Bay 2 to
0x06 Bay 6

Example: Which hub bay is the T-Cube unit fitted

RX 66, 00, 06, 00, 01, 50

MGMSG_RACK_REQ_STATUSBITS 0x0226
MGMSG_RACK_GET_STATUSBITS 0x0227

This method is applicable only to the MMR modular rack, and 2- and 3-channel card slot type controllers such as the BSC103 and BPC202.

Function: The USER IO connector on the rear panel of these units exposes several digital inputs. This function returns several status flags pertaining to the status of the inputs on the rack modules, or the motherboard of the controller unit hosting the single channel controller card.
These flags are returned in a single 32-bit integer parameter and can provide additional useful status information for client application development. The individual bits (flags) of the 32-bit integer value are described below.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
26	02	Status Bits	00	d	s

GET:
Response structure (10 bytes)
6-byte header followed by 4-byte data packet as follows:

0	1	2	3	4	5	7	8	9	10
header						*Data*			
27	02	04	00	d|	s	StatusBits			

Data Structure:

field	description	format
StatusBits	The status bits for the associated controller channel. The meaning of the individual bits (flags) of the 32-bit integer value will depend on the controller and are described in the following table.	dword

Hex Value	Bit Number	Description
0x00000001	1	Digital output 1 state (1 - logic high, 0 - logic low).
0x00000002	2	Digital output 2 state (1 - logic high, 0 - logic low).
0x00000004	3	Digital output 3 state (1 - logic high, 0 - logic low).
0x00000008	4	Digital output 4 state (1 - logic high, 0 - logic low).

Example: With destination being 0x11 (motherboard – see Introduction) and bay being bay 1, slot 2 (0x22)

TX 27, 02, 04, 00, 01, 22, 00, 00, 00, 00

Header: 27, 02, 04, 00, 01, 22: GetStatusBits, 04 byte data packet, bay 1 slot 2.

MGMSG_RACK_SET_DIGOUTPUTS 0x0228
MGMSG_RACK_REQ_DIGOUTPUTS 0x0229
MGMSG_RACK_GET_DIGOUTPUTS 0x0230

This method is applicable only to the MMR rack modules, and 2- and 3-channel card slot type controllers such as the BSC103 and BPC202.

Function: The USER IO connector on the rear panel of these units exposes several digital outputs. These functions set and return the status of the outputs on the rack modules, or the motherboard of the controller unit hosting the single channel controller card.
These flags are returned in a single 32-bit integer parameter and can provide additional useful status information for client application development. The individual bits (flags) of the 32-bit integer value are described below.

SET:
Data structure (6 bytes)

0	1	2	3	4	5
header only					
28	02	Dig OP	00	d	s

Hex Value	Bit Number	Description
0x00000001	1	Digital output 1 state (1 - logic high, 0 - logic low).
0x00000002	2	Digital output 2 state (1 - logic high, 0 - logic low).
0x00000004	3	Digital output 3 state (1 - logic high, 0 - logic low).
0x00000008	4	Digital output 4 state (1 - logic high, 0 - logic low).

Example: With destination being 0x11 (motherboard – see Introduction) and bay being bay 1, slot 2 (0x22), set Digital output 1 high

TX 28, 02, 01, 22, 11, 01,

Header: 28, 02, 01, 22, 11, 01: SetDigOutputs, 01 OP1 High, bay 1 slot 2, d=motherboard, s=PC.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
29	02	00	00	d	s

GET:
Response structure (6 bytes)

0	1	2	3	4	5
header only					
30	02	00	00	d	s

See SET above for structure

MGMSG_MOD_SET_DIGOUTPUTS 0x0213
MGMSG_MOD_REQ_DIGOUTPUTS 0x0214
MGMSG_MOD_GET_DIGOUTPUTS 0x0215

Function: The CONTROL IO connector on the rear panel of the unit exposes several digital outputs. The number of outputs available depends on the type of unit. This message is used to configure these digital outputs.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	02	Bit	00	d	s

Note. On brushless DC controllers (e.g., BBD201), the digital output and trigger output use a common pin. Before calling this message to set the digital output, the trigger functionality must be disabled by calling the Set Trigger message.

The outputs are set (and returned) in the bits of the Bits parameter, input No 1 being the least significant bit and input No 4 being the most significant. The number of bits used is dependent on the number of digital outputs present on the associated hardware unit.

For example, to turn on the digital output on a BSC201 motor controller, the least significant bit of the Bits parameter should be set to 1. Similarly, to turn on all four digital outputs on a BNT001 NanoTrak unit, the bits of the Bits parameter should be set to 1111 (15), and to turn the same outputs off, the Bits should be set to 0000.

Example: Set the digital input of the BSC201 controller on:

TX 13, 02, 01, 00, 50, 01

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
14	02	Bits	00	d	s

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
15	02	Bit	00	d	s

For structure see SET message above.

MGMSG_HW_SET_KCUBEMMILOCK 0x0250
MGMSG_HW_REQ_KCUBEMMILOCK 0x0251
MGMSG_HW_GET_KCUBEMMILOCK 0x0252

THIS MESSAGE IS APPLICABLE ONLY TO K-CUBE NanoTrak (KNA101-IR), K-Cube Laser Source (KLS1550 and KLS635) and K-Cube Laser Diode Driver (KLD101) UNITS

Function: This message is used to lock/unlock the controls on the top panel of the K-Cube units (wheel, joystick, buttons etc). Safety features such as the power switch and laser enable are not affected by this message. The message has global effect for all channels present on a particular unit. If the MMILock byte is set to 0x01, the controls are locked, if set to 0x02 the controls are unlocked. This message is non-volatile and will reset to unlock with each power cycle.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
50	02	00	MMILock	d	s

Example: Lock the top panel controls:

TX 50, 02, 00, 01, 50, 01

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
51	02	00	MMILock	d	s

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
52	02	00	MMILock	d	s

For structure see SET message above.

MGMSG_RESTOREFACTORYSETTINGS 0x0686

THIS MESSAGE IS APPLICABLE ONLY TO THE FOLLOWING CONTROLLERS:
Benchtop Piezo Controllers (BPC301 and BPC303)
K-CUBE NanoTrak (KNA101-IR)
K-Cube Laser Source (KLS1550 and KLS635)
K-Cube Laser Diode Driver (KLD101) UNITS

Function: If the system has become unstable, possibly due to multiple changes to parameter values, this message can be sent to the controller to reset parameters to the default values stored in the EEPROM.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
86	06	Chan Ident	00	d	s

Motor Control Messages

Introduction

The ‘Motor’ messages provide the functionality required for a client application to control one or more of the Thorlabs series of motor controller units. This range of motor controllers covers DC servo and stepper drivers in a variety of formats including compact Cube type controllers, benchtop units and 19" rack based modular drivers. Note for ease of description, the TSC001 T-Cube Solenoid Controller is considered here as a motor controller. The list of controllers covered by the motor messages includes:

BSC001 – 1 Channel Benchtop Stepper Driver
BSC002 – 2 Channel Benchtop Stepper Driver
BMS001 – 1 Channel Benchtop Low Power Stepper Driver
BMS002 – 2 Channel Benchtop Low Power Stepper Driver
MST601 – 2 Channel Modular Stepper Driver
MST602 – 2 Channel Modular Stepper Driver (2013 onwards)
BSC101 – 1 Channel Benchtop Stepper Driver (2006 onwards)
BSC102 – 2 Channel Benchtop Stepper Driver (2006 onwards)
BSC103 – 3 Channel Benchtop Stepper Driver (2006 onwards)
BSC201 – 1 Channel Benchtop Stepper Driver (2012 onwards)
BSC202 – 2 Channel Benchtop Stepper Driver (2012 onwards)
BSC203 – 3 Channel Benchtop Stepper Driver (2012 onwards)
BBD101 – 1 Channel Benchtop Brushless DC Motor Driver
BBD102 – 2 Channel Benchtop Brushless DC Motor Driver
BBD103 – 3 Channel Benchtop Brushless DC Motor Driver
BBD201 – 1 Channel Benchtop Brushless DC Motor Driver
BBD202 – 2 Channel Benchtop Brushless DC Motor Driver
BBD203 – 3 Channel Benchtop Brushless DC Motor Driver
OST001 – 1 Channel Cube Stepper Driver
ODC001 – 1 Channel Cube DC Servo Driver
TST001 – 1 Channel T-Cube Stepper Driver
TDC001 – 1 Channel T-Cube DC Servo Driver
TSC001 – 1 Channel T-Cube Solenoid Driver
TDIxxx – 2 Channel Brushless DC Motor Driver
TBD001 – 1 Channel T-Cube Brushless DC Driver
KST101 – 1 Channel K-Cube Stepper Driver
KDC101 – 1 Channel K-Cube DC Servo Driver
KSC101 – 1 Channel K-Cube Solenoid Driver
KBD101 – 1 Channel K-Cube Brushless DC Driver

The motor messages can be used to perform activities such as homing stages, absolute and relative moves, changing velocity profile settings and operation of the solenoid state (on solenoid control units). With a few exceptions, these messages are generic and apply equally to both single and dual channel units.
Where applicable, the target channel is identified in the Chan Ident parameter and on single channel units, this must be set to CHAN1_ID. On dual channel units, this can be set to CHAN1_ID, CHAN2_ID or CHANBOTH_ID as required.

For details on the operation of the motor controller, and information on the principles of operation, refer to the handbook supplied with the unit.

MGMSG_HW_YES_FLASH_PROGRAMMING 0x0017

Function: This message is sent by the server on start-up; however, it is a deprecated message (i.e., has no function) and can be ignored.

Command structure (6 bytes):

0	1	2	3	4	5
header only					
17	00	Unused	Unused	d	s

REQUEST: N/A

MGMSG_HW_NO_FLASH_PROGRAMMING 0x0018

Function: This message is sent on start up to notify the controller of the source and destination addresses. A client application must send this message as part of its initialization process.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
18	00	00	00	d	s

REQUEST: N/A
GET: N/A

MGMSG_MOT_SET_POSCOUNTER 0x0410
MGMSG_MOT_REQ_POSCOUNTER 0x0411
MGMSG_MOT_GET_POSCOUNTER 0x0412

Function: Used to set the 'live' position count in the controller. In general, this command is not normally used. Instead, the stage is homed immediately after power-up (at this stage the position is unknown as the stage is free to move when the power is off); and after the homing process is completed the position counter is automatically updated to show the actual position. From this point onwards the position counter always shows the actual absolute position.

SET:
Command structure (12 bytes)
6-byte header followed by 6-byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
10	04	06	00	d|	s	Chan Ident		Position			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Position	The new value of the position counter as a 32-bit signed integer, encoded in the Intel format. The scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the position counter for channel 2 to 10.0 mm

TX 10, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00

Header: 10, 04, 06, 00, A2, 01: SetPosCounter, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Position: 40, 0D, 03, 00: Set Counter to 10 mm (10 x 20,000)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6-byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
12	04	06	00	d|	s	Chan Ident		Position			

For structure see SET message above.

MGMSG_MOT_SET_ENCCOUNTER 0x0409
MGMSG_MOT_REQ_ENCCOUNTER 0x040A
MGMSG_MOT_GET_ENCCOUNTER 0x040B

Function: Similarly, to the PosCounter message described previously, this message is used to set the encoder count in the controller and is only applicable to stages and actuators fitted with an encoder. In general, this command is not normally used. Instead, the stage is homed immediately after power-up (at this stage the position is unknown as the stage is free to move when the power is off); and after the homing process is completed the position counter is automatically updated to show the actual position. From this point onwards the encoder counter always shows the actual absolute position.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
09	04	06	00	d|	s	Chan Ident		Encoder Count			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Encoder Count	The new value of the encoder counter as a 32-bit signed integer, encoded in the Intel format. The scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the encoder counter for channel 2 to 10.0 mm

TX 09, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00

Header: 09, 04, 06, 00, A2, 01: SetEncCounter, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Position: 40, 0D, 03, 00: Set Counter to 10 mm (10 x 20,000)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
0A	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
0B	04	06	00	d|	s	Chan Ident		Encoder Count			

For structure see SET message above.

MGMSG_MOT_SET_VELPARAMS 0x0413
MGMSG_MOT_REQ_VELPARAMS 0x0414
MGMSG_MOT_GET_VELPARAMS 0x0415

Function: Used to set the trapezoidal velocity parameters for the specified motor channel. For DC servo controllers, the velocity is set in encoder counts/sec and acceleration is set in encoder counts/sec/sec.
For stepper motor controllers the velocity is set in microsteps/sec and acceleration is set in microsteps/sec/sec.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
13	04	0E	00	d|	s	Chan Ident		Min Velocity			

12	13	14	15	16	17	18	19
Data							
Acceleration				Max Velocity			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Minimum (Start) Vel	The minimum (start) velocity in encoder counts/sec Currently, this 4-byte value is always zero	long
Acceleration	The acceleration in encoder counts /sec/sec. 4-byte unsigned long value. If applicable, the scaling between real time values and this parameter is detailed in Section 8.	long
Maximum Vel	The maximum (final) velocity in encoder counts /sec. 4-byte unsigned long value. If applicable, the scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the trapezoidal velocity parameters for chan 2 as follows:

Min Vel: zero
Acceleration: 10 mm/sec/sec
Max Vel: 99 mm/sec

TX 13, 04, 0E, 00, A2, 01, 01, 00, 00, 00, 00, 00, B0, 35, 00, 00, CD, CC, CC, 00

Header: 13, 04, 0E, 00, A2, 01: Set Vel Params, 0EH (14) byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Min Vel: *00, 00, 00, 00:* Set min velocity to zero
Accel: *89, 00, 00, 00:* Set acceleration to 10 mm/sec/sec (13.744 x 10)
Max Vel: *9E, C0, CA, 00:* Set max velocity to 99 mm/sec (134218 x 99)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
14	04	Chan Ident	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
15	04	0E	00	d|	s	Chan Ident		Min Velocity			

12	13	14	15	16	17	18	19
Data							
Acceleration				Max Velocity			

For structure see SET message above.

MGMSG_MOT_SET_JOGPARAMS 0x0416
MGMSG_MOT_REQ_JOGPARAMS 0x0417
MGMSG_MOT_GET_JOGPARAMS 0x0418

Function: Used to set the velocity jog parameters for the specified motor channel, For DC servo controllers, values set in encoder counts. For stepper motor controllers the values is set in microsteps.

SET:

Command structure (28 bytes)

6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
16	04	16	00	d|	s	Chan Ident		Jog Mode		Jog Step Size	

12	13	14	15	16	17	18	19	20	21
Data									
Jog Step Size		Jog Min Velocity				Jog Acceleration			

22	23	24	25	26	27
Data					
Jog Max Velocity				Stop Mode	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Jog Mode	This 2-byte value can be 1 for continuous jogging or 2 for single step jogging. In continuous jogging mode the movement continues for as long as the jogging trigger (the jogging button on the GUI or an external signal) is being active. In single step mode triggering jogging initiates a single move whose step size is defined as the next parameter (see below).	word
Jog Step Size	The jog step size in encoder counts. The scaling between real time values and this parameter is detailed in Section 8.	long
Jog Min Velocity	The minimum (start) velocity in encoder counts /sec. Currently, this 4-byte value is always zero.	long
Jog Acceleration	The acceleration in encoder counts /sec/sec The scaling between real time values and this parameter is detailed in Section 8.	long
Jog Max Velocity	The maximum (final) velocity in encoder counts /sec. The scaling between real time values and this parameter is detailed in Section 8.	long
Jog Stop Mode	The stop mode. This 16-bit word can be 1 for immediate (abrupt) stop or 2 for profiled stop (with controlled deceleration).	word

Example: MLS203 and BBD102: Set the jog parameters for channel 2 as follows:
Jog Mode: Continuous
Jog Step Size:0.05 mm
Jog Min Vel: Zero
Jog Accel: 10 mm/sec/sec
Jog Max Vel: 99 mm/sec
Jog Stop Mode: Profiled

TX 16, 04, 16, 00, A2, 01, 01, 00, 01, 00, E8, 03, 00, 00, 00, 00, 00, 00, B0,35, 00, 00, CD, CC, CC, 00, 02, 00

Header: 16, 04, 16, 00, A2, 01: Set Jog Params, 16H (28) byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Jog Mode: *01,00,:* Set jog mode to 'continuous'
Jog Step Size: *E8, 03, 00, 00:* Set jog step size to 0.05 mm (1,000 encoder counts).
Jog Min Vel: 00, 00, 00, 00: Set min jog velocity to zero
Jog Accel: *89, 00, 00, 00:* Set acceleration to 10 mm/sec/sec (13.744 x 10)
Jog Max Vel: *9E, C0, CA, 00:* Set max velocity to 99 mm/sec (134218 x 99)
Jog Stop Mode: 02, 00: Set jog stop mode to 'Profiled Stop'.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
17	04	Chan Ident	00	d	s

GET:
Response structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
18	04	16	00	d|	s	Chan Ident		Jog Mode		Jog Step Size	

12	13	14	15	16	17	18	19	20	21
Data									
Jog Step Size		Jog Min Velocity				Jog Acceleration			

22	23	24	25	26	27
Data					
Jog Max Velocity				Stop Mode	

For structure see SET message above.

MGMSG_MOT_REQ_ADCINPUTS 0x042B
MGMSG_MOT_GET_ADCINPUTS 0x042C

Function: This message reads the voltage applied to the analog input on the rear panel CONTROL IO connector and returns a value in the ADCInput1 parameter. The returned value is in the range 0 to 32768, which corresponds to zero to 5 V.
Note. The ADCInput2 parameter is not used at this time.
In this way, a 0 to 5V signal generated by a client system could be read in by calling this method and monitored by a custom client application. When the signal reaches a specified value, the application could instigate further actions, such as a motor move.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
2B	04	Chan Ident	00	d	s

GET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
2C	04	04	00	d|	s	ADCInput1		ADCInput2	

Data Structure:

field	description	format
ADCInput1	The voltage state of the analog input pin, in the range 0 to 32768, which corresponds to zero to 5 V.	word
ADCInput2	Not used	word

Example: Get the ADC input state

RX 2C, 04, 04, 00, A2, 01, 01, 00, 00, 00,

Header: 2B, 04, 04, 00, A2, 01: GetADCInputs, 04 byte data packet, Channel 2.
ADCInput1: 00, 80: ADC Input 1 = 5V
ADCInput2: 00, 00: Not Used r

MGMSG_MOT_SET_POWERPARAMS 0x0426
MGMSG_MOT_REQ_POWERPARAMS 0x0427
MGMSG_MOT_GET_POWERPARAMS 0x0428

Note for BSC20x, MST602 and TST101 controller users
If the controllers listed above are used with Thorlabs SoftwareServer, the ini file will typically have values set of 5 for the rest power and 30 for the move power. Although these values are loaded when the server boots only the rest power value is used. This allows the user to set the rest current as normal. The move power however is not used. The move power is set within the controller as a function of velocity. This command can be used only to set the rest power.
The command MGMSG_MOT_REQ_POWERPARAMS will return the default values or the values that were set.

Function: The power needed to hold a motor in a fixed position is much smaller than that required for a move. It is good practice to decrease the power in a stationary motor to reduce heating, and thereby minimize thermal movements caused by expansion. This message sets a reduction factor for the rest power and the move power values as a percentage of full power. Typically, move power should be set to 100% and rest power to a value significantly less than this.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
26	04	06	00	d|	s	Chan Ident		RestFactor		MoveFactor	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
RestFactor	The phase power value when the motor is at rest, in the range 1 to 100 (i.e., 1% to 100% of full power).	word
MoveFactor	The phase power value when the motor is moving, in the range 1 to 100 (i.e., 1% to 100% of full power).	word

Example: Set the phase powers for channel 2 for TST001 unit

TX 26, 04, 06, 00, A2, 01, 01, 00, 0A, 00, 64, 00

Header: 26, 04, 06, 00, A2, 01: SetPowerParams, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TST001)
RestFactor: 0A, 00: Set rest power to 10% of full power
MoveFactor: 64, 00: Set move power to 100% of full power

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
27	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
28	04	06	00	d|	s	Chan Ident		RestFactor		MoveFactor	

For structure see SET message above.

MGMSG_MOT_SET_GENMOVEPARAMS 0x043A
MGMSG_MOT_REQ_GENMOVEPARAMS 0x043B
MGMSG_MOT_GET_GENMOVEPARAMS 0x043C

Function: Used to set the general move parameters for the specified motor channel. Currently this refers specifically to the backlash settings.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
3A	04	06	00	d|	s	Chan Ident		Backlash Distance			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Backlash Distance	The value of the backlash distance as a 4-byte signed integer, which specifies the relative distance in position counts. The scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the backlash distance for chan 2 to 1 mm:

TX 3A, 04, 06, 00, A2, 01, 01, 00, 20, 4E, 00, 00,

Header: 3A, 04, 06, 00, A2, 01: SetGenMoveParams, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Backlash Dist: *20, 4E, 00, 00:* Set backlash distance to 1 mm (20,000 encoder counts).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
3B	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
3C	04	06	00	d|	s	Chan Ident		Backlash Distance			

For structure see SET message above.

MGMSG_MOT_SET_MOVERELPARAMS 0x0445
MGMSG_MOT_REQ_MOVERELPARAMS 0x0446
MGMSG_MOT_GET_MOVERELPARAMS 0x0447

Function: Used to set the relative move parameters for the specified motor channel. The only significant parameter currently is the relative move distance itself. This gets stored by the controller and is used the next time a relative move is initiated.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
45	04	06	00	d|	s	Chan Ident		Relative Distance			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Relative Distance	The distance to move. This is a 4-byte signed integer that specifies the relative distance in position encoder counts. The scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the relative move distance for chan 2 to 10 mm:

TX 45, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00,

Header: 45, 04, 06, 00, A2, 01: SetMoveRelParams, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Rel Dist: *40, 0D, 03, 00:* Set relative move distance to 10 mm (10 x 20,000 encoder counts).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
46	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
47	04	06	00	d|	s	Chan Ident		Relative Distance			

For structure see SET message above.

MGMSG_MOT_SET_MOVEABSPARAMS 0x0450
MGMSG_MOT_REQ_MOVEABSPARAMS 0x0451
MGMSG_MOT_GET_MOVEABSPARAMS 0x0452

Function: Used to set the absolute move parameters for the specified motor channel. The only significant parameter currently is the absolute move position itself. This gets stored by the controller and is used the next time an absolute move is initiated.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
50	04	06	00	d|	s	Chan Ident		Absolute Position			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Absolute Position	The absolute position to move. This is a 4 byte signed integer that specifies the absolute position in position encoder counts. The scaling between real time values and this parameter is detailed in Section 8.	long

Example: MLS203 and BBD102: Set the absolute move position for chan 2 to 10 mm:

TX 50, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00,

Header: 50, 04, 06, 00, A2, 01: SetMoveAbsParams, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Abs Pos: *40, 0D, 03, 00:* Set absolute move position to 10 mm (200,000 encoder counts).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
51	04	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
52	04	06	00	d|	s	Chan Ident		Absolute Position			

For structure see SET message above.

MGMSG_MOT_SET_HOMEPARAMS 0x0440
MGMSG_MOT_REQ_HOMEPARAMS 0x0441
MGMSG_MOT_GET_HOMEPARAMS 0x0442

Function: Used to set the home parameters for the specified motor channel. These parameters are stage specific and for the MLS203 stage implementation the only parameter that can be changed is the homing velocity.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
40	04	0E	00	d|	s	Chan Ident		Home Dir		Limit Switch	

12	13	14	15	16	17	18	19
Data							
Home Velocity				Offset Distance			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Home Direction	The direction sense for a move to Home, either 1 - forward/Positive or 2 - reverse/negative.	word
Limit Switch	The limit switch associated with the home position 1 - hardware reverse or 4 - hardware forward	word
Home Velocity	The homing velocity. A 4 byte unsigned long value. The scaling between real time values and this parameter is detailed in Section 8.	long
Offset Distance	The distance of the home position from the Home Limit Switch. This is a 4 byte signed integer that specifies the offset distance in position encoder counts. The scaling between real time values and this parameter is detailed in Section 8	long

Example: MLS203 and BBD102: Set the home parameters for chan 2 as follows:
Home Direction: Not used (always positive).
Limit Switch: Not used
Home Vel: 24 mm/sec
Offset Dist: Not used.

TX 40, 04, 0E, 00, A2, 01, 01, 00, 00, 00, 00, 00, 33. 33, 33, 00, 00, 00, 00, 00

Header: 40, 04, 0E, 00, A2, 01: SetHomeParams, 14 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Home Direction: *00, 00:* Not Applicable
Limit Switch: 00, 00: Not Applicable
Home Velocity: 33, 33, 33, 00: 24 mm/sec (3355443/134218)
Offset Distance: 00, 00, 00, 00: Not used

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	04	Chan Ident	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
42	04	0E	00	d|	s	Chan Ident		Home Dir		Limit Switch	

12	13	14	15	16	17	18	19
Data							
Home Velocity				Offset Distance			

For structure see SET message above.

MGMSG_MOT_SET_LIMSWITCHPARAMS 0x0423
MGMSG_MOT_REQ_LIMSWITCHPARAMS 0x0424
MGMSG_MOT_GET_LIMSWITCHPARAMS 0x0425

These functions are not applicable to BBD10x units.

Function: Used to set the limit switch parameters for the specified motor channel.

SET:
Command structure (22 bytes)
6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
23	04	10	00	d|	s	Chan Ident		CW Hardlimit		CCW Hardlimit	

12	13	14	15	16	17	18	19	20	21
Data									
CW Soft Limit				CCW Soft Limit				Limit Mode	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
CW Hard Limit	The operation of the Clockwise hardware limit switch when contact is made. 0x01 Ignore switch or switch not present. 0x02 Switch makes on contact. 0x03 Switch breaks on contact. 0x04 Switch makes on contact - only used for homes (e.g. limit switched rotation stages). 0x05 Switch breaks on contact - only used for homes (e.g. limit switched rotations stages). 0x06 For PMD based brushless servo controllers only - uses index mark for homing. Note. Set upper bit to swap CW and CCW limit switches in code. Both CWHardLimit and CCWHardLimit structure members will have the upper bit set when limit switches have been physically swapped. 0x80 // bitwise OR'd with one of the settings above.	word
CCW Hard Limit	The operation of the counter clockwise hardware limit switch when contact is made.	word
CW Soft Limit	Clockwise software limit in position steps. A 32 bit unsigned long value, the scaling factor between real time values and this parameter is 1 mm is equivalent to 134218. For example, to set the clockwise software limit switch to 100 mm, send a value of 13421800. **(Not applicable to TDC001 units)**	long
CCW Soft Limit	Counter clockwise software limit in position steps (scaling as for CW limit). **(Not applicable to TDC001 units)**	long

Software Limit Mode	Software limit switch mode 0x01 Ignore Limit 0x02 Stop Immediate at Limit 0x03 Profiled Stop at limit 0x80 Rotation Stage Limit (bitwise OR'd with one of the settings above) **(Not applicable to TDC001 units)**	word

Example: Set the limit switch parameters for chan 2 as follows:
CW Hard Limit – switch makes.
CCW Hard Limit - switch makes
CW Soft Limit – set to 100 mm
CCW Soft Limit - set to 0 mm
Software Limit Mode – Profiled Stop

TX 23, 04, 10, 00, A2, 01, 01, 00, 02, 00, 02, 00, E8. CC, CC, 00, 00, 00, 00, 00, 03, 00

Header: 23, 04, 10, 00, A2, 01: SetLimSwitchParams, 16 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
CW Hard Limit: *02, 00:* Switch Makes
CCW Hard Limit: 02, 00: Switch Makes
CW Soft Limit: E8, CC, CC, 00: 100 mm (13421800/134218)
CCW Soft Limit: 00, 00, 00, 00: 0 mm
Soft Limit Mode: 03, 00: Profiled Stop at Limit

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
24	04	Chan Ident	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
25	04	10	00	d|	s	Chan Ident		CW Hardlimit		CCW Hardlimit	

12	13	14	15	16	17	18	19	20	21
Data									
CW Soft Limit				CCW Soft Limit				Limit Mode	

For structure see SET message above.

MGMSG_MOT_MOVE_HOME 0x0443
MGMSG_MOT_MOVE_HOMED 0x0444

Function: Sent to start a home move sequence on the specified motor channel (in accordance with the home parameters above).

TX structure (6 bytes):

0	1	2	3	4	5
header only					
43	04	Chan Ident	0x	d	s

Example: Home the motor channel in bay 2

TX 43, 04, 01, 00, 22, 01

HOMED:

Function: No response on initial message, but upon completion of home sequence controller sends a “homing completed” message:

RX structure (6 bytes):

0	1	2	3	4	5
header only					
44	04	Chan Ident	0x	d	s

Example: The motor channel in bay 2 has been homed

RX 44, 04, 01, 00, 01, 22

MGMSG_MOT_MOVE_RELATIVE 0x0448

Function: This command can be used to start a relative move on the specified motor channel (using the relative move distance parameter above). There are two versions of this command: a shorter (6-byte header only) version and a longer (6 byte header plus 6 data bytes) version. When the first one is used, the relative distance parameter used for the move will be the parameter sent previously by a MGMSG_MOT_SET_MOVERELPARAMS command. If the longer version of the command is used, the relative distance is encoded in the data packet that follows the header.

Short version:

TX structure (6 bytes):

0	1	2	3	4	5
header only					
48	04	Chan Ident	0x	d	s

Example: Move the motor associated with channel 2 by 10 mm. (10 mm was previously set in the MGMSG_ MOT_SET_MOVERELPARAMS method).

TX 48, 04, 01, 00, 22, 01

Long version:
The alternative way of using this command is by appending the relative move params structure (MOT_SET_MOVERELPARAMS) to this message header.

Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
48	04	06	00	d|	s	Chan Ident		Relative Distance			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	Word
Relative Distance	The distance to move. This is a 4 byte signed integer that specifies the relative distance in position encoder counts. In the BBD10X series controllers the encoder resolution is 20,000 counts per mm, therefore, to set a relative move distance of 1 mm, set this parameter to 20,000 (twenty thousand).	Long

Example: Move the motor associated with chan 2 by 10 mm:

TX 48, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00,

Header: 45, 04, 06, 00, A2, 01: MoveRelative, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Rel Dist: *40, 0D, 03, 00:* Set absolute move distance to 10 mm (200,000 encoder counts).

Upon completion of the relative move the controller sends a Move Completed message as described following.

MGMSG_MOT_MOVE_COMPLETED 0x0464

Function: No response on initial message, but upon completion of the relative or absolute move sequence, the controller sends a "move completed" message:

RX structure (20 bytes):

0	1	2	3	4	5
header only					
64	04	Chan Ident	0x	d|	s

Followed by a 14-byte data packet described by the same status structures (i.e. MOTSTATUS and MOTDCSTATUS) described in the STATUS UPDATES section that follows.

MGMSG_MOT_MOVE_ABSOLUTE 0x0453

Function: Used to start an absolute move on the specified motor channel (using the absolute move position parameter above). As previously described in the "MOVE RELATIVE" command, there are two versions of this command: a shorter (6-byte header only) version and a longer (6 byte header plus 6 data bytes) version. When the first one is used, the absolute move position parameter used for the move will be the parameter sent previously by a MGMSG_MOT_SET_MOVEABSPARAMS command. If the longer version of the command is used, the absolute position is encoded in the data packet that follows the header.

Short version:

TX structure (6 bytes):

0	1	2	3	4	5
header only					
53	04	Chan Ident	0x	d	s

Example: Move the motor associated with channel 2 to 10 mm. (10 mm was previously set in the MGMSG_ MOT_SET_MOVEABSPARAMS method).

TX 53, 04, 01, 00, 22, 01

Long version:
The alternative way of using this command by appending the absolute move params structure (MOTABSMOVEPARAMS) to this message header.

Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
53	04	06	00	d|	s	Chan Ident		Absolute Distance			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	Word
Absolute Distance	The distance to move. This is a 4 byte signed integer that specifies the absolute distance in position encoder counts. In the BBD10X series controllers the encoder resolution is 20,000 counts per mm, therefore, to set an absolute move distance of 100 mm, set this parameter to 2,000,000 (two million).	Long

Example: Move the motor associated with chan 2 to 10 mm:

TX 53, 04, 06, 00, A2, 01, 01, 00, 40, 0D, 03, 00,

Header: 45, 04, 06, 00, A2, 01: MoveAbsolute, 06 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Abs Dist: *40, 0D, 03, 00:* Set the absolute move distance to 10 mm (200,000 encoder counts).

Upon completion of the absolute move the controller sends a Move Completed message as previously described.

MGMSG_MOT_MOVE_JOG 0x046A

Function: Sent to start a jog move on the specified motor channel.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
6A	04	Chan Ident	Direction	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Direction	The direction to Jog. Set this byte to 0x01 to jog forward, or to 0x02 to jog in the reverse direction.	word

Upon completion of the jog move the controller sends a Move Completed message as previously described.

Note. The direction of the jog move is device dependent, i.e., on some devices jog forward may be towards the home position while on other devices it could be the opposite.

MGMSG_MOT_MOVE_VELOCITY 0x0457

Function: This command can be used to start a move on the specified motor channel.
When this method is called, the motor will move continuously in the specified direction, using the velocity parameters set in the MGMSG_MOT_SET_VELPARAMS command until either a stop command (either StopImmediate or StopProfiled) is called, or a limit switch is reached.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
57	04	Chan Ident	Direction	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Direction	The direction to Jog. Set this byte to 0x01 to move forward, or to 0x02 to move in the reverse direction.	word

Upon completion of the move the controller sends a Move Completed message as previously described.

Example: Move the motor associated with channel 2 forwards.

TX 57, 04, 01, 01, 22, 01

Special Note for MST602 units

The MST602 is a true 2-channel controller, rather than two single channel controllers. In this case, as well as the Chan Ident parameter, the channel being addressed is also specified in the Direction parameter (byte 3). The lower 4 bit nibble of the direction parameter is used to address channel 1 and the upper 4 bit nibble is used to address channel 2.

Examples
to move channel1 forward, TX 57, 04, 01, 01,22,01
to move channel 1 backward, TX 57, 04, 01, 02,22,01

to move channel 2 forward, TX 57, 04, 02, 10,22,01
to move channel 2 backward, TX 57, 04, 02, 20,22,01

MGMSG_MOT_MOVE_STOP 0x0465

Function: Sent to stop any type of motor move (relative, absolute, homing or move at velocity) on the specified motor channel.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
65	04	Chan Ident	Stop Mode	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Stop Mode	The stop mode defines either an immediate (abrupt) or profiles tops. Set this byte to 0x01 to stop immediately, or to 0x02 to stop in a controller (profiled) manner.	word

Upon completion of the stop move the controller sends a Move Stopped message as described following

MGMSG_MOT_MOVE_STOPPED 0x0466

Function: No response on initial message, but upon completion of the stop move, the controller sends a “move stopped” message:

RX structure (20 bytes):

0	1	2	3	4	5
header only					
66	04	0E	0x	d|	s

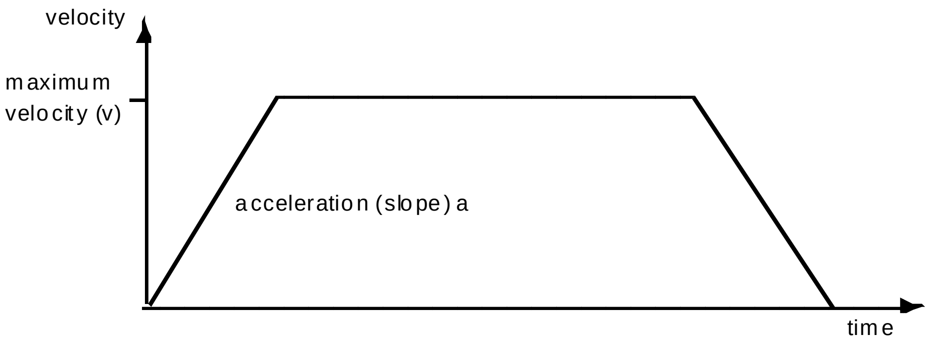
Followed by a 14-byte data packet described by the same status structures (i.e., MOTSTATUS and MOTDCSTATUS) described in the STATUS UPDATES section that follows.

MGMSG_MOT_SET_BOWINDEX 0x04F4
MGMSG_MOT_REQ_BOWINDEX 0x04F5
MGMSG_MOT_GET_BOWINDEX 0x04F6

Function:

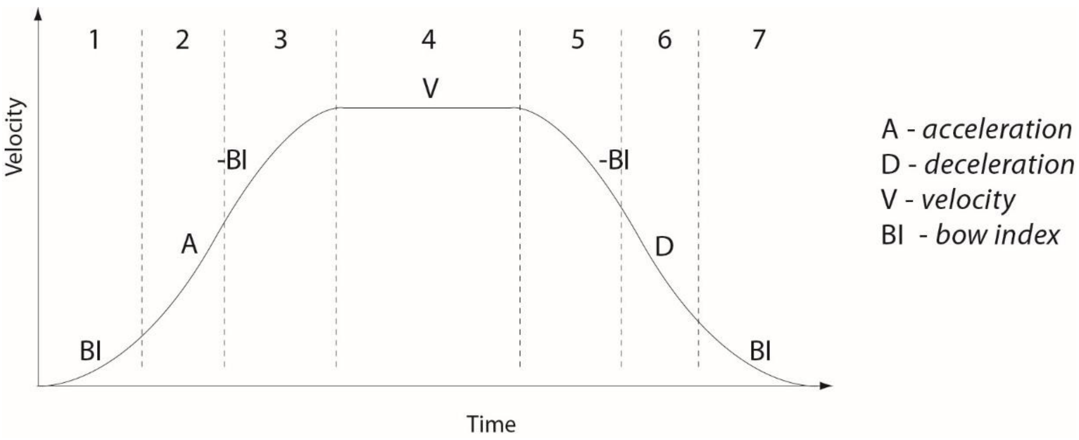
To prevent the motor from stalling, it must be ramped up gradually to its maximum velocity. Certain limits to velocity and acceleration result from the torque and speed limits of the motor, and the inertia and friction of the parts it drives. The system incorporates a trajectory generator, which performs calculations to determine the instantaneous position, velocity, and acceleration of each axis at any given moment. During a motion profile, these values will change continuously. Once the move is complete, these parameters will then remain unchanged until the next move begins. The specific move profile created by the system depends on several factors, such as the profile mode and profile parameters presently selected, and other conditions such as whether a motion stop has been requested.

The Bow Index parameter is used to set the profile mode to either Trapezoidal or S-curve. A Bow Index of '0' selects a trapezoidal profile. An index value of '1' to '18' selects an S-curve profile. In either case, the velocity and acceleration of the profile are specified using the Velocity Profile parameters on the Moves/Jogs tab. The Trapezoidal profile is a standard, symmetrical acceleration/deceleration motion curve, in which the start velocity is always zero. This profile is selected when the Bow Index field is set to '0'.



In a typical trapezoidal velocity profile, (see above), the stage is ramped at acceleration 'a' to a maximum velocity 'v'. As the destination is approached, the stage is decelerated at 'a' so that the final position is approached slowly in a controlled manner.

The S-curve profile is a trapezoidal curve with an additional 'Bow Value' parameter, which limits the rate of change of acceleration and smooths out the contours of the motion profile. The Bow Value is applied in mm/s^3 and is derived from the Bow Index as follows:
Bow Value = 2 $^{(\text{Bow Index} -1)}$ within the range 1 to 262144 (Bow Index 1 to 18).
In this profile mode, the acceleration increases gradually from 0 to the specified acceleration value, then decreases at the same rate until it reaches 0 again at the specified velocity. The same sequence in reverse brings the axis to a stop at the programmed destination position.



Example
The figure above shows a typical S-curve profile. In segment (1), the S-curve profile drives the axis at the specified Bow Index (BI) until the maximum acceleration (A) is reached. The axis continues to accelerate linearly (Bow Index = 0) through segment (2). The profile then applies the negative value of Bow Index to reduce the acceleration to 0 during segment (3). The axis is now at the maximum velocity (V), at which it continues through segment (4). The profile then decelerates in a similar manner to the acceleration phase, using the Bow Index to reach the maximum deceleration (D) and then bring the axis to a stop at the destination.

Note
The higher the Bow Index, then the shorter the BI phases of the curve, and the steeper the acceleration and deceleration phases. High values of Bow Index may cause a move to overshoot.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
F4	04	04	00	d|	s	Chan Ident		Bow Index	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
BowIndex	This parameter is used to set the profile mode to either Trapezoidal or S-curve. A Bow Index of '0' selects a trapezoidal profile. An index value of '1' to '18' selects an S-curve profile.	word

Example: Set the Bow Index to 18 for Channel 1 as follows:

TX F4, 04, 04, 00, A2, 01, 01, 00, 12, 00,

Header: F4, 04, 04, 00, A2, 01: Set_BowIndex, 04 byte data packet,
Chan Ident: 01, 00: Channel 1
Bow Index: *12, 00,*: Set the Bow Index to 18

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
F5	04	Chan Ident	00	d	s

GET:

6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
F6	04	04	00	d|	s	Chan Ident		Bow Index	

For structure see SET message above.

MGMSG_MOT_SET_DCPIDPARAMS 0x04A0
MGMSG_MOT_REQ_DCPIDPARAMS 0x04A1
MGMSG_MOT_GET_DCPIDPARAMS 0x04A2

Function: Used to set the position control loop parameters for the specified motor channel.
The motion processor within the controller uses a position control loop to determine the motor command output. The purpose of the position loop is to match the actual motor position and the demanded position. This is achieved by comparing the demanded position with the actual position to create a position error, which is then passed through a digital PID-type filter. The filtered value is the motor command output.
NOTE. These settings apply to LM628/629 based servo controllers (only TDC001 at this time). Refer to data sheet for National Semiconductor LM628/LM629 for further details on setting these PID related parameters.

SET:
Command structure (26 bytes)
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
A0	04	14	00	d|	s	Chan Ident		Proportional			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Integral				Derivative				Integral Limit			

24	25
Data	
FilterControl	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Proportional	The proportional gain. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767.	long
Integral	The integral gain. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767.	long
Derivative	The derivative gain. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 32767.	long
Integral Limit	The Integral Limit parameter is used to cap the value of the Integrator to prevent runaway of the integral sum at the output. It accepts values in the range 0 to 32767. If set to 0 then the integration term in the PID loop is ignored.	long
FilterControl	Identifies which of the above parameters are applied by	word

	setting the corresponding bit to '1'. By default, all parameters are applied, and this parameter is set to 0F (1111).	

Example: Set the PID parameters for TDC001 as follows:
Proportional: 65
Integral: 175
Derivative: 600
Integral Limit: 20,000
FilCon: 15

TX A0, 04, 14, 00, D0, 01, 01, 00, 41, 00, AF, 00, 58, 02, 20, 4E, 00, 00, 0F, 00

Header: A0, 04, 14, 00, D0, 01: Set_DCPIDParams, 20 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Proportional: *41, 00,:* Set the proportional term to 65
Integral: *AF, 00,:* Set the integral term to 175
Derivative: 58, 02,: Set the derivative term to 600
Integral Limit: 20, 4E, 00, 00,: Set the integral limit to 20,000
FilterControl: 0F, 00: Set all terms to active.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
A1	04	Chan Ident	00	d	s

GET:
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
A2	04	14	00	d|	s	Chan Ident		Proportional			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Integral				Derivative				Integral Limit			

24	25
Data	
FilterControl	

For structure see Set message above.

MGMSG_MOT_SET_AVMODES 0x04B3
MGMSG_MOT_REQ_AVMODES 0x04B4
MGMSG_MOT_GET_AVMODES 0x04B5

It is not applicable for K10CR1 and K10CR2

Function: The LED on the control keypad can be configured to indicate certain driver states.
All modes are enabled by default. However, it is recognised that in a light sensitive environment, stray light from the LED could be undesirable. Therefore, it is possible to enable selectively, one or all the LED indicator modes described below by setting the appropriate value in the Mode Bits parameter.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
B3	04	04	00	d|	s	Chan Ident		ModeBits	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
ModeBits	The mode of operation for the LED is set according to the hex value entered in the mode bits. 1 LEDMODE_IDENT: The LED will flash when the 'Ident' message is sent. 2 LEDMODE_LIMITSWITCH: The LED will flash when the motor reaches a forward or reverse limit switch. 8 LEDMODE_MOVING: The LED is lit when the motor is moving.	word

Example: Set the LED to flash when the IDENT message is sent, and when the motor is moving.

TX B3, 04, 04, 00, D0, 01, 01, 00, 09, 00,

Header: B3, 04, 04, 00, D0, 01: SetAVModes, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
ModeBits: 09, 00 (i.e. 1 + 8)

Similarly, if the ModeBits parameter is set to '11' (1 + 2 + 8) all modes will be enabled.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	04	Chan Ident	00	d	s

GET:
Response structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
B5	04	04	00	d|	s	Chan Ident		ModeBits	

For structure see SET message above.

MGMSG_MOT_SET_POTPARAMS 0x04B0
MGMSG_MOT_REQ_POTPARAMS 0x04B1
MGMSG_MOT_GET_POTPARAMS 0x04B2

Function: The potentiometer slider on the control panel is sprung, such that when released it returns to its central position. In this central position the motor is stationary. As the slider is moved away from the centre, the motor begins to move; the speed of this movement increases as the slider deflection is increased. Bidirectional control of motor moves is possible by moving the slider in both directions. The speed of the motor increases by discrete amounts rather than continuously, as a function of slider deflection. These speed settings are defined by 4 pairs of parameters. Each pair specifies a pot deflection value (in the range 0 to 127) together with an associated velocity (set in encoder counts/sec) to be applied at or beyond that deflection. As each successive deflection is reached by moving the pot slider, the next velocity value is applied. These settings are applicable in either direction of pot deflection, i.e., 4 possible velocity settings in the forward or reverse motion directions.
Note. The scaling factor between encoder counts and mm/sec depends on the specific stage/actuator being driven.

SET:
Command structure (32 bytes)
6 byte header followed by 26 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
B0	04	1A	00	d|	s	Chan Ident		ZeroWnd		Vel1	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Vel1		Wnd1		Vel2				Wnd2		Vel3	

24	25	26	27	28	29	30	31
Data							
Vel3		Wnd3		Vel4			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
ZeroWnd	The deflection from the mid position (in ADC counts 0 to 127) before motion can start	word
Vel1	The velocity (in encoder counts /sec) to move when between Wnd0 and PotDef1	long
Wnd1	The deflection from the mid position (in ADC counts, Wnd0 to 127) to apply Vel1	word
Vel2	The velocity (in encoder counts /sec) to move when between PotDef1 and PotDef2	long
Wnd2	The deflection from the mid position (in ADC counts, PotDef1 to 127) to apply Vel2	word

Vel3	The velocity (in encoder counts/sec) to move when between PotDef2 and PotDef3	long
Wnd3	The deflection from the mid position (in ADC counts PotDef2 to 127) to apply Vel3	word
Vel4	The velocity (in encoder counts /sec) to move when beyond PotDef3	long

Example: For the Z8 series motors, there are 512 encoder counts per revolution of the motor. The output shaft of the motor goes into a 67:1 planetary gear head. This requires the motor to rotate 67 times to rotate the 1.0 mm pitch lead screw one revolution. The result is the lead screw advances by 1.0 mm.
Therefore, a 1 mm linear displacement of the actuator is given by

512 x 67 = 34,304 encoder counts

whereas the linear displacement of the lead screw per encoder count is given by

1.0 mm / 34,304 counts = 2.9 x 10-5 mm (29 nm).

Typical parameters settings Hex (decimal)

ZeroWnd – 14 (20)
Vel1 – 66, 0D,00,00 (3430)
Wnd1 – 32 (50)
Vel2 – CC, 1A, 00, 00 (6860)
Wnd2 – 50 (80)
Vel3 – 32, 28, 00, 00 (10290)
Wnd3 – 64 (100)
Vel4 – 00, 43, 00, 00 (17152)

Using the parameters above, no motion will start until the pot has been deflected to 20 (approx 1/6 full scale deflection), when the motor will start to move at 0.1mm/sec. At a deflection of 50 (approx 2/5 full scale deflection) the motor velocity will increase to 0.2mm/sec, and at 80, velocity will increase to 0.3 mm/sec. When the pot is deflected to 100 and beyond, the velocity will be 0.5 mm/sec.

Note. It is acceptable to set velocities equal to reduce the number of speeds, however this is not allowed for the deflection settings, whereby the Wnd3 Pot Deflection value must be greater than Wnd2 Pot Deflection value.

TX *B0, 04, 1A, 00, D0, 01,* 01, 00, 01, 00, E8, 03, 00, 00, 00, 00, 00, 00, B0,35, 00, 00, CD, CC, CC, 00, 02, 00

Header: B0, 04, 1A, 00, D0, 01: Set Pot Params, 1AH (26) byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Wnd0: 14 (20 ADC Counts)
Vel1: 66, 0D,00,00 (3430 Encoder Counts/sec = 0.1 mm/sec)
PotDef1: 32 (50 ADC Counts)
Vel2: CC, 1A, 00, 00 (6860 Encoder Counts/sec = 0.2 mm/sec)
PotDef2: 50 (80 ADC Counts)

Vel3: 32, 28, 00, 00 (10290 Encoder Counts/sec = 0.3 mm/sec)
PotDef3: 64 (100 ADC Counts)
Vel4: 00, 43, 00, 00 (17152 Encoder Counts/sec = 0.5 mm/sec)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
17	04	Chan Ident	00	d	s

GET:
Response structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
B0	04	1A	00	d|	s	Chan Ident		ZeroWnd		Vel1	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Vel1		Wnd1		Vel2				Wnd2		Vel3	

24	25	26	27	28	29	30	31
Data							
Vel3		Wnd3		Vel4			

For structure see SET message above.

MGMSG_MOT_SET_BUTTONPARAMS 0x04B6
MGMSG_MOT_REQ_BUTTONPARAMS 0x04B7
MGMSG_MOT_GET_BUTTONPARAMS 0x04B8

Function: The control keypad can be used either to jog the motor, or to perform moves to absolute positions. This function is used to set the front panel button functionality.

SET:

Command structure (22 bytes)

6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
B6	04	10	00	d|	s	Chan Ident		Mode		Position1	

12	13	14	15	16	17	18	19	20	21
Data									
Position1		Position2				TimeOut1		TimeOut2	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Mode	The buttons on the keypad can be used either to jog the motor (jog mode), or to perform moves to absolute positions (go to position mode). If set to 0x01, the buttons are used to jog the motor. Once set to this mode, the move parameters for the buttons are taken from the 'Jog' parameters set via the 'Move/Jogs' settings tab or the SetJogParams methods. If set to 0x02, each button can be programmed with a different position value (as set in the Position 1 and Position 2 parameters), such that the controller will move the motor to that position when the specific button is pressed.	word
Position1	The position (in encoder counts) to which the motor will move when the top button is pressed. This parameter is applicable only if 'Go to Position is selected in the 'Mode' parameter.	long
Position2	The position (in encoder counts) to which the motor will move when the bottom button is pressed. This parameter is applicable only if 'Go to Position is selected in the 'Mode' parameter.	long
TimeOut1	A 'Home' move can be performed by pressing and holding both buttons. Furthermore, the present position can be entered into the Position 1 or Position 2 parameter by holding down the associated button. The Time Out parameter specifies the time in ms that button 1 must be depressed. This function is independent of the 'Mode' setting and in normal circumstances should not require adjustment. **(Not applicable to TDC001 units)**	word
TimeOut2	As TimeOut1 but for Button 2.	word

Example: Set the button parameters for TDC001 as follows:
Mode: Go To Position
Position1: 0.5 mm
Position2: 1.2 mm
TimeOut: 2 secs

TX B6, 04, 10, 00, D0, 01, 01, 00, 02, 00, C0, 12, 00, 00, 00, 00, 00, 00, 00, 00

Header: B6, 04, 10, 00, D0, 01: SetButtonParams, 10H (16) byte data packet, Generic USB Device
Chan Ident: 01, 00: Channel 1 (always set to 1 for TDC001)
Mode: 02, 00 (i.e. Go to position)
Position1: 00, 43, 00, 00 (17152 Encoder Counts = 0.5 mm)
Position2: CC, A0, 00, 00 (41164 encoder counts = 1.2 mm):
TimeOut: D0, 07: (2 seconds)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
B7	04	Chan Ident	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
B8	04	10	00	d|	s	Chan Ident		Mode		Position1	

12	13	14	15	16	17	18	19	20	21
Data									
Position1		Position2				TimeOut1		TimeOut2	

For structure see SET message above.

MGMSG_MOT_SET_EEPROMPARAMS 0x04B9

Function: Used to save the parameter settings for the specified message. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the 'Settings' button found in the lower right hand corner of the user interface).

SET:

Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
B9	04	04	00	d|	s	Chan Ident		MsgID	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
MsgID	The message ID of the message containing the parameters to be saved.	word

Example:

TX B9, 04, 04, 00, D0, 01, 01, 00, B6, 04,

Header: B9, 04, 04, 00, D0, 01: Set_EEPROMPARAMS, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
MsgID: Save parameters specified by message 04B6 (SetButtonParams).

MGMSG_MOT_SET_POSITIONLOOPPARAMS 0x04D7
MGMSG_MOT_REQ_POSITIONLOOPPARAMS 0x04D8
MGMSG_MOT_GET_POSITIONLOOPPARAMS 0x04D9

Function: Used to set the position control loop parameters for the specified motor channel.
The motion processors within the BBD series controllers use a position control loop to determine the motor command output. The purpose of the position loop is to match the actual motor position and the demanded position. This is achieved by comparing the demanded position with the actual encoder position to create a position error, which is then passed through a digital PID-type filter. The filtered value is the motor command output.

SET:
Command structure (34 bytes)
6 byte header followed by 28 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D7	04	1C	00	d|	s	Chan Ident		Kp Pos		Integral	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
ILimPos				Derivative		KdTimePos		KoutPos		KvffPos	

24	25	26	27	28	29	30	31	32	33
Data									
KaffPos		PosErrLim				ParamSetIx		N/A	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Kp Pos	The proportional gain. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767.	word
Integral	The integral gain. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767.	word
ILimPos	The Integral Limit parameter is used to cap the value of the Integrator to prevent runaway of the integral sum at the output. It accepts values in the range 0 to 7FFFFFFF. If set to 0 then the integration term in the PID loop is ignored.	dword
Derivative	The derivative gain. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 32767.	word
KdTimePos	Under normal circumstances, the derivative term of the PID loop is recalculated at every servo cycle. However, it may be desirable to reduce the sampling rate to a lower value, in order to increase stability or simplify tuning. The KdTimePos parameter is used to set the sampling rate. For example, if	word

	set to 10, the derivative term is calculated every 10 servo cycles. The value is set in cycles, in the range 1 to 32767.	
KoutPos	The KoutPos parameter is a scaling factor applied to the output of the PID loop. It accepts values in the range 0 to 65535, where 0 is 0% and 65535 is 100%.	word
KvffPos	The KvffPos and KaffPos parameters are velocity and acceleration feed-forward terms that are added to the output of the PID filter to assist in tuning the motor drive signal. They accept values in the range 0 to 32767.	word
KaffPos		word
PosErrLim	Under certain circumstances, the actual encoder position may differ from the demanded position by an excessive amount. Such a large position error is often indicative of a potentially dangerous condition such as motor failure, encoder failure or excessive mechanical friction. To warn of, and guard against this condition, a maximum position error can be set in the PosErrLim parameter, in the range 0 to 7FFFFFFF. The actual position error is continuously compared against the limit entered, and if exceeded, the Motion Error bit (bit 15) of the Status Register is set and the associated axis is stopped.	dword
ParamSetIx	It is possible to enter a set of PID parameters for different operating scenarios, e.g. motor is stationary, motor is accelerating, motor is at constant velocity. The specific set of PID parameters to use when the function is called is set in the ParamSetIx parameter as follows: 0 = Position PID parameters to apply when motor is stationary 1 = Position PID parameters to apply when motor is accelerating 2 = Position PID parameters to apply when motor is at constant velocity **NOTE**. This parameter is not applicable to BBD10x and BBD20x units and in this case, the units use the values from the last time the command was sent.	word
Not Used		word

Example: Set the PID parameters for chan 2 as follows:
Proportional: 65
Integral: 175
Integral Limit: 80,000
Derivative: 600
KdTimePos: 5
KoutPos: 5%
KvffPos: 0
KaffPos: 1000
PosErrLim: 65535
ParamSetIx: 1

TX D7, 04, 1C, 00, A2, 01, 01, 00, 41, 00, AF, 00, 80, 38, 01, 00, 58, 02, 05, 00, CD, 0C, 00, 00, E8, 03, FF, FF, 01, 00, 00, 00

Header: D7, 04, 1C, 00, A2, 01: Set_PositionLoopParams, 28 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Proportional: *41, 00,:* Set the proportional term to 65
Integral: *AF, 00,:* Set the integral term to 175
Integral Limit: 80, 38, 01, 00,: Set the integral limit to 80,000
Derivative: 58, 02,: Set the derivative term to 600
KdTimePos: 05, 00,: Set the sampling rate to 5 cycles
KoutPos: CD, 0C,: Set the output scaling factor to 5% (i.e. 3277)
KvffPos: 00, 00,: Set the velocity feed forward value to zero
KaffPos: E8, 03,: Set the acceleration feed forward value to 1000
PosErrLim: FF, FF, 00, 00,: Set the position error limit to 65535.
ParamSetIx: 01, 00,: Use PID parameter set 1.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
D8	04	Chan Ident	00	d	s

GET:
Response structure (34 bytes)
6 byte header followed by 28 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D9	04	1C	00	d|	s	Chan Ident		Kp Pos		Integral	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
ILinPos				Derivatve		KdTimePos		KoutPos		KvffPos	

24	25	26	27	28	29	30	31	32	33
Data									
KaffPos		PosErrLim				N/A		N/A	

For structure see SET message above.

MGMSG_MOT_SET_MOTOROUTPUTPARAMS 0x04DA
MGMSG_MOT_REQ_MOTOROUTPUTPARAMS 0x04DB
MGMSG_MOT_GET_MOTOROUTPUTPARAMS 0x04DC

Function: Used to set certain limits that can be applied to the motor drive signal. The individual limits are described below.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
DA	04	0E	00	d|	s	Chan Ident		Cont Current Lim		Energy Limit	

12	13	14	15	16	17	18	19
Data							
Motor Limit		Motor Bias		Not Used		Not Used	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
ContCurrentLim	The system incorporates a current 'foldback' facility, whereby the continuous current level can be capped. The continuous current limit is set in the ContCurrentLim parameter, which accepts values as a percentage of maximum peak current, in the range 0 to 32767 (0 to 100%), which is the default maximum level set at the factory (this maximum value cannot be altered).	word
EnergyLim	When the current output of the drive exceeds the limit set in the ContCurrentLim parameter, accumulation of the excess current energy begins. The EnergyLim parameter specifies a limit for this accumulated energy, as a percentage of the factory set default maximum, in the range 0 to 32767 (0 to 100%). When the accumulated energy exceeds the value specified in the EnergyLim parameter, a 'current foldback' condition is said to exist, and the commanded current is limited to the value specified in the ContCurrentLim parameter. When this occurs, the Current Foldback status bit (bit 25) is set in the Status Register. When the accumulated energy above the ContCurrentLim value falls to 0, the limit is removed and the status bit is cleared.	word
MotorLim	The MotorLim parameter sets a limit for the motor drive signal and accepts values in the range 0 to 32767 (100%). If the system produces a value greater than the limit set, the motor command takes the limiting value. For example, if MotorLim is set to 30000 (91.6%), then signals greater than 30000 will be output as 30000 and values less than -30000 will be output as -30000.	word
MotorBias	Not implemented.	word

Not Used		word
Not Used		word

Example: Set the motor output parameters for chan 2 as follows:
Continuous Current: 20%
Energy Limit: 14%
Motor Limit: 100%
Motor Bias: zero

TX DA, 04, 0E, 00, A2, 01, 01, 00, 99, 19, C0, 12, 00, 00, 00, 00, 00, 00, 00, 00

Header: DA, 04, 0E, 00, A2, 01: Set MotorOutputParams, 0EH (14) byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Cont Current Limit:
Energy Limit: 99, 19: Set the energy limit to 14%
Motor Limit: C0, 12: Set the motor limit to 100%
Motor Bias: 00, 00: Set the motor bias to zero

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
DB	04	Chan Ident	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
DC	04	0E	00	d|	s	Chan Ident		Cont Current Lim		Energy Limit	

12	13	14	15	16	17	18	19
Data							
Motor Limit		Motor Bias		Not Used		Not Used	

For structure see SET message above.

MGMSG_MOT_SET_TRACKSETTLEPARAMS 0x04E0
MGMSG_MOT_REQ_TRACKSETTLEPARAMS 0x04E1
MGMSG_MOT_GET_TRACKSETTLEPARAMS 0x04E2

Function: Moves are generated by an internal profile generator, and are based on either a trapezoidal or S-curve trajectory. A move is considered complete when the profile generator has completed the calculated move and the axis has 'settled' at the demanded position. This command contains parameters which specify when the system is settled.

Further Information

The system incorporates a monitoring function, which continuously indicates whether or not the axis has ‘settled’. The ‘Settled’ indicator is bit 14 in the Status Register and is set when the associated axis is settled. Note that the status bit is controlled by the processor, and cannot be set or cleared manually.

The axis is considered to be ‘settled’ when the following conditions are met:

* the axis is at rest (i.e. not performing a move),

* the error between the demanded position and the actual motor position is less than or equal to a specified number of encoder counts (0 to 65535) set in the *SettleWnd* parameter (Settle Window),

* the above two conditions have been met for a specified number of cycles (settle time, 1 cycle = 102.4 µs), set in the *SettleTime* parameter (range 0 to 32767).

The above settings are particularly important when performing a sequence of moves. If the PID parameters are set such that the settle window cannot be reached, the first move in the sequence will never complete, and the sequence will stall. The settle window and settle time values should be specified carefully, based on the required positional accuracy of the application. If positional accuracy is not a major concern, the settle time should be set to '0'. In this case, a move will complete when the motion calculated by the profile generator is completed, irrespective of the actual position attained, and the settle parameters described above will be ignored.

The processor also provides a ‘tracking window’, which is used to monitor servo performance outside the context of motion error. The tracking window is a programmable position error limit within which the axis must remain, but unlike the position error limit set in the SetDCPositionLoopParams method, the axis is not stopped if it moves outside the specified tracking window. This function is useful for processes that rely on the motor’s correct tracking of a set trajectory within a specific range. The tracking window may also be used as an early warning for performance problems that do not yet qualify as motion error.

The size of the tracking window (i.e. the maximum allowable position error while remaining within the tracking window) is specified in the *TrackWnd* parameter, in the range 0 to 65535. If the position error of the axis exceeds this value, the Tracking Indicator status bit (bit 13) is set to 0 in the Status Register. When the position error returns to within the window boundary, the status bit is set to 1.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E0	04	0C	00	d|	s	Chan Ident		Time		Settle Window	

12	13	14	15	16	17
Data					
Track Window		Not Used		Not Used	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Time	The time that the associated axis must be settled before the 'Settled' status bit is set. The time is set in cycles, in the range 0 to 32767, 1 cycle = 102.4 µs.	word
Settle Window	The position error is defined as the error between the demanded position and the actual motor position. This parameter specifies the number of encoder counts (in the range 0 to 65535) that the position error must be less than or equal to, before the axis is considered 'settled'.	word
Track Window	The maximum allowable position error (in the range 0 to 65535) whilst tracking.	word
Not Used		word
Not Used		word

Example: Set the track and settle parameters for chan 2 as follows:
Settle Time: 20%
Settle Window: 14%
Track Window: 100%

s
TX E0, 04, 0C, 00, A2, 01, 01, 00, 00, 00, 14, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: E0, 04, 0C, 00, A2, 01: SetTrackSettledParams, 0CH (12) byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Time: 00, 00: Set the Settle time to zero
Settle Window: 14, 00: Set the settle window to 20 encoder counts
Track Window: 00, 00: Set the track window to zero.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
E1	04	Chan Ident	00	d	s

GET:
Response structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E2	04	0C	00	d|	s	Chan Ident		Time		Settle Window	

12	13	14	15	16	17
Data					
Track Window		Not Used		Not Used	

For structure see SET message above.

MGMSG_MOT_SET_PROFILEMODEPARAMS 0x04E3
MGMSG_MOT_REQ_PROFILEMODEPARAMS 0x04E4
MGMSG_MOT_GET_PROFILEMODEPARAMS 0x04E5

Function: The system incorporates a trajectory generator, which performs calculations to determine the instantaneous position, velocity and acceleration of each axis at any given moment. During a motion profile, these values will change continuously. Once the move is complete, these parameters will then remain unchanged until the next move begins.
The specific move profile created by the system depends on several factors, such as the profile mode and profile parameters presently selected, and other conditions such as whether a motion stop has been requested. This method is used to set the profile mode to either 'Trapezoidal' or 'S-curve'.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E3	04	0C	00	d|	s	Chan Ident		Mode		Jerk	

12	13	14	15	16	17
Data					
Jerk		Not Used		Not Used	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Mode	The move profile to be used: Trapezoidal: 0 S-Curve: 2 The Trapezoidal profile is a standard, symmetrical acceleration/deceleration motion curve, in which the start velocity is always zero. The S-curve profile is a trapezoidal curve with an additional 'Jerk' parameter, which limits the rate of change of acceleration and smooths out the contours of the motion profile. In this profile mode, the acceleration increases gradually from 0 to the specified acceleration value, then decreases at the same rate until it reaches 0 again at the specified velocity. The same sequence in reverse brings the axis to a stop at the programmed destination position.	word
Jerk	The Jerk value is specified in mm/s^3 in the Jerk parameter, and accepts values in the range 0 to 4294967295. It is used to specify the maximum rate of change in acceleration in a single cycle of the basic trapezoidal curve. 1.0 mm/s^3 is equal to 92.2337 jerk units.	dword
Not Used		word
Not Used		word

Example: Set the profile mode parameters for chan 2 as follows:
Profile Mode: S-curve
Jerk: 10,000 mm^3

TX E3, 04, 0C, 00, A2, 01, 01, 00, 02, 00, E1, 12, 0E, 00, 00, 00, 00, 00,

Header: E3, 04, 0C, 00, A2, 01: Set ProfileModeParams, 0CH (12) byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Profile Mode: 02, 00: Set the profile mode to S-Curve
Jerk: E1, 12,0E, 00: Set the jerk value to 10,000 mm/sec^3 (i.e. 922337)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
E4	04	Chan Ident	00	d	s

GET:
Response structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E5	04	0C	00	d|	s	Chan Ident		Mode		Jerk	

12	13	14	15	16	17
Data					
Jerk		Not Used		Not Used	

For structure see SET message above.

MGMSG_MOT_SET_JOYSTICKPARAMS 0x04E6
MGMSG_MOT_REQ_JOYSTICKPARAMS 0x04E7
MGMSG_MOT_GET_JOYSTICKPARAMS 0x04E8

Function: The MJC001 joystick console has been designed for use by microscopists to provide intuitive, tactile, manual positioning of the stage. The console consists of a two axis joystick for XY control which features both low and high gear modes. This message is used to set max velocity and acceleration values for these modes.

SET:
Command structure (26 bytes)
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E6	04	14	00	d|	s	Chan Ident		JSGearLowMaxVel			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JSGearHighMaxVel				JSGearHighLowAccn				JSGearHighHighAccn			

24	25
Data	
DirSense	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
JSGearLowMaxVel	Specifies the max velocity (in encoder counts/cycle) of a joystick move when low gear mode is selected. It accepts values in the range 0 to 4294967295. 1 mm / sec equals 134218 PMD units	long
JSGearHighMaxVel	Specifies the max velocity (in encoder counts/cycle) of a joystick move when high gear mode is selected. It accepts values in the range 0 to 4294967295. 1 mm / sec equals 134218 PMD units	long
JSGearLowAccn	Specifies the acceleration (in encoder counts/cycle) of a joystick move when low gear mode is selected. It accepts values in the range 0 to 4294967295. 1 mm /sec^2 equals 13.7439 PMD units.	long
JSGearHighAccn	Specifies the acceleration (in encoder counts/cycle) of a joystick move when high gear mode is selected. It accepts values in the range 0 to 4294967295. 1 mm /sec^2 equals 13.7439 PMD units.	long
DirSense	The actual direction sense of any joystick initiated move is dependent upon the application. This parameter can be used to reverse the sense of direction for a particular application and is useful when matching joystick direction sense to actual stage direction sense. DIRSENSE_POS 0X0001 Direction Positive DIRSENSE_NEG 0X0002 Direction Negative	word

Example: Set the joystick parameters for bay 2 as follows:
JSGearLowMaxVel: 1 mm/sec
JSGearHighMaxVel: 10 mm/sec
JSGearLowAccn: 0.5 mm /sec^2
JSGearHighAccn: 5.0 mm /sec^2
DirSens: Positive

TX E6, 04, 14, 00, A2, 01, 01, 00, 4A, 0C, 02, 00, E4, 7A, 14, 00, 07, 00, 00, 00, 46, 00, 00, 00, 01, 00

Header: E6, 04, 14, 00, A2, 01: SetJoystickParams, 14H (20) byte data packet, bay 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
*JSGearLowMaxVel: 4A, 0C, 02, 00 (*134218)
JSGearHighMaxVel: E4, 7A, 14, 00 (1342180)
JSGearLowAccn: 07, 00, 00, 00 (7.0)
JSGearHighAccn: 46, 00, 00, 00 (70.0)
DirSens: *01, 00*

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
E7	04	Chan Ident	00	d	s

GET:
Response structure (26 bytes)
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E8	04	14	00	d|	s	Chan Ident		JSGearLowMaxVel			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JSGearHighMaxVel				JSGearHighLowAccn				JSGearHighHighAccn			

24	25
Data	
DirSense	

For structure see SET message above.

MGMSG_MOT_SET_CURRENTLOOPPARAMS 0x04D4
MGMSG_MOT_REQ_CURRENTLOOPPARAMS 0x04D5
MGMSG_MOT_GET_CURRENTLOOPPARAMS 0x04D6

Function: Used to set the current control loop parameters for the specified motor channel.
The motion processors within the BBD series controllers use digital current control as a technique to control the current through each phase winding of the motors. In this way, response times are improved and motor efficiency is increased. This is achieved by comparing the required (demanded) current with the actual current to create a current error, which is then passed through a digital PI-type filter. The filtered current value is used to develop an output voltage for each motor coil.
This method sets various constants and limits for the current feedback loop.

SET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D4	04	12	00	d|	s	Chan Ident		Phase		KpCurrent	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
KiCurrent		ILimCurrent		DeadBand		Kff		ParamSetIx		Not Used	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Phase	The current phase to set: PHASEA 0 PHASEB 1 PHASEA AND B 2	word
KpCurrent	The proportional gain. Together with the KiCurrent this term determines the system response characteristics and accept values in the range 0 to 32767.	word
KiCurrent	The integral gain. Together with the KpCurrent this term determines the system response characteristics and accept values in the range 0 to 32767.	word
ILimCurrent	The ILimCurrent parameter is used to cap the value of the Integrator to prevent runaway of the integral sum at the output. It accepts values in the range 0 to 32767. If set to 0 then the integration term in the PID loop is ignored.	word
IDeadBand	The IDeadBand parameter allows an integral dead band to be set, such that when the error is within this dead band, the integral action stops, and the move is completed using the proportional term only. It accepts values in the range 0	word

	to 32767.	
Kff	The Kff parameter is a feed-forward term that is added to the output of the PID filter to assist in tuning the motor drive signal. It accepts values in the range 0 to 32767.	word
ParamSetIx	It is possible to enter a set of PID parameters for different operating scenarios, e.g. motor is stationary, motor is in motion or not yet settled at target position. The specific set of PID parameters to use when the function is called is set in the ParamSetIx parameter as follows: 0 = Normal current loop parameter set (motor in motion, or not yet settled at target position) 1 = Settled current loop parameter set (motor stationary, settled at target position) **NOTE**. This parameter is not applicable to BBD10x and BBD20x units and in this case, the units use the values from the last time the command was sent.	word
Not Used		word

Example: Set the limit switch parameters for chan 2 as follows:
Phase: A and B
KpCurrent: 35
KiCurrent: 80
ILimCurrent: 32,767
DeadBand: 50
Kff: 0
ParamSetIx: 1

TX D4, 04, 12, 00, A2, 01, 01, 00, 02, 00, 23, 00, 50, 00, FF, 7F, 32, 00, 00, 00, 01, 00, 00, 00,

Header: D4, 04, 12, 00, A2, 01: Set_CurrentLoopParams, 18 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Phase: 02, 00: Set Phase A and Phase B
KpCurrent: *23, 00,:* Set the proportional term to 35
KiCurrent: *50, 00,:* Set the integral term to 80
ILimCurrent: FF, 7F,: Set the integral limit to 32767
IDeadBand: 32, 00,: Set the deadband to 50
Kff: 00, 00: Set the feed forward value to zero
ParamSetIx: 01, 00 Use parameter set 1.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
D8	04	Chan Ident	00	d	s

GET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D6	04	12	00	d|	s	Chan Ident		Phase		KpCurrent	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
KiCurrent		ILimCurrent		DeadBand		Kff		Not Used		Not Used	

For structure see SET message above.

MGMSG_MOT_SET_SETTLEDCURRENTLOOPPARAMS 0x04E9
MGMSG_MOT_REQ_SETTLEDCURRENTLOOPPARAMS 0x04EA
MGMSG_MOT_GET_SETTLEDCURRENTLOOPPARAMS 0x04EB

Function: These commands assist in maintaining stable operation and reducing noise at the demanded position. They allow the system to be tuned such that errors caused by external vibration and manual handling (e.g. loading of samples) are minimized, and are applicable only when the stage is settled, i.e. the Axis Settled status bit (bit 14) is set.

SET:

Command structure (24 bytes)

6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E9	04	12	00	d|	s	Chan Ident		Phase		KpSettled	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
KiSettled		ILimSettled		DeadBandSet		KffSettled		Not Used		Not Used	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Phase	The current phase to set: PHASEA 0 PHASEB 1 PHASEA AND B 2	word
KpSettled	The proportional gain. Together with the KiSettled this term determines the system response characteristics and accept values in the range 0 to 32767.	word
KiSettled	The integral gain. Together with the KpSettled this term determines the system response characteristics and accept values in the range 0 to 32767.	word
ILimSettled	The ILimSettled parameter is used to cap the value of the Integrator to prevent runaway of the integral sum at the output. It accepts values in the range 0 to 32767. If set to 0 then the integration term in the PID loop is ignored.	word
IDeadBandSettled	The IDeadBandSettled parameter allows an integral dead band to be set, such that when the error is within this dead band, the integral action stops, and the move is completed using the proportional term only. It accepts values in the range 0 to 32767.	word
KffSettled	The KffSettled parameter is a feed-forward term that is added to the output of the PID filter to assist in tuning the motor drive signal. It accepts values in the range 0 to 32767.	word
Not Used		word
Not Used		word

Example: Set the limit switch parameters for chan 2 as follows:
Phase: A and B
KpSettled: 0
KiSettled: 40
ILimSettled: 30,000
DeadBandSettled: 50
KffSettled:500

TX E9, 04, 12, 00, A2, 01, 01, 00, 02, 00, 00, 00, 28, 00, 30, 75, 32, 00, F4, 01, 00, 00, 00, 00,

Header: D4, 04, 12, 00, A2, 01: Set_SettledCurrentLoopParams, 18 byte data packet, Channel 2.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Phase: 02, 00: Set Phase A and Phase B
KpCurrent: *00, 00,:* Set the proportional term to zero
KiCurrent: *28, 00,:* Set the integral term to 40
ILimCurrent: 30, 75,: Set the integral limit to 30,000
IDeadBand: 32, 00,: Set the deadband to 50
Kff: F4, 01: Set the feed forward value to 500

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
D8	04	Chan Ident	00	d	s

GET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
EB	04	12	00	d|	s	Chan Ident		Phase		KpSettled	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
KiSettled		ILimSettled		DeadBandSet		KffSettled		Not Used		Not Used	

For structure see SET message above.

MGMSG_MOT_SET_STAGEAXISPARAMS 0x04F0
MGMSG_MOT_REQ_STAGEAXISPARAMS 0x04F1
MGMSG_MOT_GET_STAGEAXISPARAMS 0x04F2

Function: The REQ and GET commands are used to obtain various parameters pertaining to the particular stage being driven. Most of these parameters are inherent in the design of the stage and cannot be altered. The SET command can only be used to increase the Minimum position value and decrease the Maximum position value, thereby reducing the overall travel of the stage.

SET:

Command structure (80 bytes)

6 byte header followed by 74 byte data packet – see Get for structure

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
F1	04	Chan Ident	00	d	s

GET:

Command structure (80 bytes)

6 byte header followed by 74 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F2	04	4A	00	d|	s	Chan ID		Stage ID		Axis ID	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Part No/Axis											

24	25	26	27	28	29	30	31	32	33	34	35
Data											
Part No/Axis				Serial Number				Counts per Unit			

36	37	38	39	40	41	42	43	44	45	46	47
Data											
MinPos				Max Pos				Max Accn			

48	49	50	51	52	53	54	55	56	57	58	59
Data											
Max Dec				Max Vel				Reserved		Reserved	

60	61	62	63	64	65	66	67	68	69	70	71
Data											
Reserved		Reserved		Reserved				Reserved			

72	73	74	75	76	77	78	79
Data							
Reserved				Reserved			

Data Structure:

field	description	format
Stage ID	This 2 byte parameter identifies the stage and axis: 00, 10 - MLS203_X_AXIS 00, 11 - MLS203_Y_AXIS	word
AxisID	Not used for the BBD series controllers	word
PartNoAxis	A 16 byte character string used to identify the stage type and axis being driven.	char
SerialNum	The Serial number of the stage	dword
CntsPerUnit	The number of encoder counts per real world unit (either mm or degrees).	dword
MinPos	The minimum position of the stage, typically zero	long
MaxPos	The maximum position of the stage in encoder counts	long
MaxAccn	The maximum acceleration of the stage in encoder counts per cycle per cycle	long
MaxDec	The maximum deceleration of the stage in encoder counts per cycle per cycle	long
MaxVel	The maximum velocity of the stage in encoder counts per cycle.	long
Reserved		word
Reserved		word
Reserved		word
Reserved		word
Reserved		dword
Reserved		dword
Reserved		dword
Reserved		dword

Example: Get the stage and axis parameters for chan 2:

RX F2, 04, 4A, 00, 81, 22, 01, 00, 11, 00, 00, 00, 4D,4C, 53, 32, 30, 33, 20, 59, 20, 41, 78, 69, 73, 00, 00, 00, 81, 96, 98, 00, 20, 4E, 00, 00, 00, 00, 00, 00, 60, E3, 16, 00, 60, 6B, 00, 00, 60, 6B, 00, 00, 9A, 99, 99, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: F2, 04, 4A, 00, 81, 22: Get_StageAxisParams, 74 byte data packet,Bay 1.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BBD202)
Stage ID: 11, 00: MLS203 Y Axis
Axis ID: *00, 00,:* Not used
PartNo Axis: *4D, 4C, 53, 32, 30, 33, 20, 59, 20, 41, 78, 69, 73, 00, 00, 00,:*
M L S 2 0 3 Y A X I S
SerialNum: 81, 96, 98, 00
CntsPerUnit 20, 4E, 00, 00: the encoder counts per unit is set to 20000
MinPos: 00, 00, 00, 00: the feed minimum position is set to zero
MaxPos: 60, E3, 16, 00: the maximum position is set to 1500000
MaxAccn: 60, 6B, 00, 00: the maximum acceleration is set to 27488
MaxDec: 60, 6B, 00, 00: the maximum deceleration is set to 27488
MaxVel: 9A, 99, 99, 01: the maximum velocity is set to 26843546

MGMSG_MOT_SET_TSTACTUATORTYPE 0x04FE

Function: This command is for use only with the TST101 driver, and is used to define an actuator type so that the TST driver knows the effective length of the stage. This information is used if a user wishes to home the stage to the far travel end. In this case, once the stage is homed the Thorlabs Software GUI count will be set to the far travel value. For example, in the case of a ZFS25 the user will see 25mm once homed. The TST holds this value as a number of Trinamic microsteps, which will be a function of the gearbox ratio, the lead screw pitch, and the motor type. So for example the number stored in the TST for the ZFS25 is 54613333.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
FE	04	Actuator Ident	00	d	s

Actuator Idents:

ZST_LEGACY_6MM	0x20
ZST_LEGACY_13MM	0x21
ZST_LEGACY_25MM	0x22
ZST_NEW_6MM	0x30
ZST_NEW_13MM	0x31
ZST_NEW_25MM	0x32
ZFS_NEW_6MM	0x40
ZFS_NEW_13MM	0x41
ZFS_NEW_25MM	0x42
DRV013_25MM	0x50
DRV014_50MM	0x51

Example: Set the actuator type to New ZFS 13 mm Travel:

Header: FE, 04, 31, 00, 50, 01:

MGMSG_MOT_GET_STATUSUPDATE 0x0481

Function: This message is returned when a status update is requested for the specified motor channel. This request can be used instead of enabling regular updates as described previously. In the BSC series controllers, each channel is seen as a separate controller with its own serial number and each card must be addressed separately.

GET:
Status update messages are received with the following format:-

Response structure (34 bytes)
6 byte header followed by 28 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
81	04	1C	00	d|	s	Chan Ident 1		Position			

12	13	14	15	16	17	18	19	20	21
Data									
EncCount				Status Bits				Chan Ident 2	

22	23	24	25	26	27	28	29	30	31	32	33
Data											
For Future Use				For Future Use				For Future Use			

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
Position	The position encoder count. In the Thorlabs Stepper Motor controllers the encoder resolution is 25,600 or 409600 counts per mm depending on the controller. Therefore a position change of 1 mm would be seen as this parameter changing by 25,600 or 409600. The LONG variable is a 32 bit value, encoded in the data stream in the Intel format.	long
EncCount	For use with encoded stages only.	long
Status Bits	The meaning of individual bits in this 32-bit variable is described in the bit mask table below (1 = active, 0 = inactive).	dword
All remaining bytes are for future use and should be ignored		

Example: Get the status update:

81, 04, 1C, 00, 81, 50, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00,

Header: 81, 04, 1C, 00, 81, 50: Get_StatusUpdate, 28 byte data packet,
Chan Ident: 01, 00: Channel 1 (always set to 1 for BSC20X)
Position: 00, 00, 00, 00:
Enc Counts: *00, 00, 00, 00:* Only used with encoded stages
Status Bits: *00, 00, 00, 00,* See below for details,:
All remaining bytes are ignored

Status Bits

bit mask	meaning
0x00000001	forward (CW) hardware limit switch is active
0x00000002	reverse (CCW) hardware limit switch is active
0x00000004	forward (CW) software limit switch is active
0x00000008	reverse (CCW) software limit switch is active
0x00000010	in motion, moving forward (CW)
0x00000020	in motion, moving reverse (CCW)
0x00000040	in motion, jogging forward (CW)
0x00000080	in motion, jogging reverse (CCW)
0x00000100	motor connected
0x00000200	in motion, homing
0x00000400	homed (homing has been completed)
0x00001000	interlock state (1 = enabled)

This is not full list of all the bits but the remaining bits reflect information about the state of the hardware that in most cases does not affect motion.

MGMSG_MOT_REQ_STATUSUPDATE 0x0480

Function: Used to request a status update for the specified motor channel. This request can be used instead of enabling regular updates as described above.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
80	04	Chan Ident	00	d	s

GET:
See previous details on MGMSG_MOT_GET_STATUSUPDATE 0x0481.

MGMSG_MOT_GET_USTATUSUPDATE 0x0491

Function: This message is returned when a status update is requested for the specified motor channel. This request can be used instead of enabling regular updates as described above.

GET:
Status update messages are received with the following format:-

Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
91	04	0E	00	d|	s	Chan Ident		Position			

12	13	14	15	16	17	18	19
Data							
Velocity		MotorCurrent		Status Bits			

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
Position	The position in encoder counts (controller units). The relationship between the encoder count and physical units such as millimetres or degrees depends on both the controller and the stage. The conversion factors are listed in Section 8: “Conversion between position, velocity and acceleration values in standard physical units and their equivalent Thorlabs parameters”. For example in the BBD20X series controllers used with the MLS203 stage, the encoder resolution is 20,000 counts per mm, therefore a position change of 1 mm would be seen as this parameter changing by 20,000 (twenty thousand). The LONG variable is a 32 bit value, encoded in the data stream in the Intel format, so for example a position of 1 million encoder counts (equivalent to 50 mm) would be sent as byte stream 0x40, 0x42, 0x0F, 0x00 since 1 million is hexadecimal 0xF4240.	long
Velocity	Actual velocity in controller units. As with position, relationship between this value and physical velocity depends on the motor and controller - see section 8 for the conversion factors. For example, this conversion factor is 204.8 for the BBD20X series controllers used with the MLS203 stage, so a real-life measured speed of 100 mm/sec is read as 205. Again, the two-byte data stream will be encoded in the Intel format.	word
Motor Current	Motor Current in mA (range -32768 to +32767). **Note**. Legacy controllers (i.e. those designed before-2020)	word

	do not return the motor current. In this case, this value is not used.	
Status Bits	The meaning of individual bits in this 32-bit variable is described below	dword

Status Bits Description

0x00000001 - P_MOT_SB_CWHARDLIMIT
0x00000002 - P_MOT_SB_CCWHARDLIMIT
Clockwise and counter-clockwise hardware limit switches. On linear stages these also correspond to the forward and reverse limit switches. (Due to the gearbox fitted in some linear stages, the clockwise and counter-clockwise directions may not match forward and reverse.)

0x00000004 - P_MOT_SB_CWSOFTLIMIT
0x00000008 - P_MOT_SB_CCWSOFTLIMIT
Clockwise and counter-clockwise software limit switches. On some controllers a software limits can be imposed on the motion, restricting it to a narrower range than the hardware limit switches.

0x00000010 - P_MOT_SB_INMOTIONCW
0x00000020 - P_MOT_SB_INMOTIONCCW
In motion, moving clockwise or counter-clockwise.

0x00000040 - P_MOT_SB_JOGGINGCW
0x00000080 - P_MOT_SB_JOGGINGCCW
Jogging, clockwise or counter-clockwise.

0x00000100 - P_MOT_SB_CONNECTED
Indicates that the motor has been recognized by the controller.

0x00000200 - P_MOT_SB_HOMING
Indicates that the motor is performing a homing move.

0x00000400 - P_MOT_SB_HOMED
Indicates that the motor has completed the homing move, the absolute position is known and therefore the position count is now valid.

0x00000800 - P_MOT_SB_INITILIZING
For 3-phase brushless motors only: the motor is performing a phase initialization procedure, attempting to establish the correct commutation phase angle. This is an essential process for brushless motors and during this process no motion related command can be responded to.

0x00001000 - P_MOT_SB_TRACKING
Actual position is within the trajectory tracking window.

0x00002000 - P_MOT_SB_SETTLED

Indicates that the motor is not moving and it is settled at the target position. The actual position has been within the target position for a specified length of time.

0x00004000 - P_MOT_SB_POSITIONERROR
Indicates that the actual position is outside the margin specified around the trajectory position. (In simple terms the motor is not where it should be.) This can occur momentarily during fast acceleration (the motor lags behind the trajectory) or when the motor is jammed, or the move is obstructed. Typically the condition can trigger the controller to disable the motor in order to prevent damage, which in turn will clear the error.
0x00008000 - P_MOT_SB_INSTRERROR
Only used on legacy controllers. Indicates that the motion controller unable to execute command received (for example, incompatible operating mode)

0x00010000 - P_MOT_SB_INTERLOCK
Used on controllers where there is a separate signal required to enable the motor.

0x00020000 - P_MOT_SB_OVERTEMP
Indicates that either the motor power driver electronics or the motor itself has reached its maximum operating temperature. Normally results in the motor drive getting disabled.

0x00040000 - P_MOT_SB_BUSVOLTFAULT
Indicates that the supply voltage to the motor is too low. Potential reasons include a power supply fault or wiring problem.

0x00080000 - P_MOT_SB_COMMUTATIONERROR
Only used for 3-phase brushless motors. Indicates a problem with the motor commutation and normally occurs if the phase initialization process has failed (see P_MOT_SB_INITILIZING). This is an unrecoverable fault that makes motion control impossible and can only be cleared by a power cycle.

0x00100000 - P_MOT_SB_DIGIP1
0x00200000 - P_MOT_SB_DIGIP2
0x00400000 - P_MOT_SB_DIGIP3
0x00800000 - P_MOT_SB_DIGIP4
Indicates the state of the digital inputs on those controllers with a limited small number of digital I/O lines. (If a controller has more than 4 digital inputs or if there are different configuration options, a separate command is used for reading the state of the input signals.)

0x01000000 - P_MOT_SB_OVERLOAD
Indicates a motor overload condition: can overcurrent condition (see P_MOT_SB_OVERCURRENT) has occurred for a long period of time and the motor has been used beyond its power handling capabilities. Normally results in the maximum output current being reduced or the motor being disabled.

0x02000000 - P_MOT_SB_ENCODERFAULT
Indicates an encoder fault in controllers that have encoder diagnostic capabilities (e.g. M30X, M30XY).

0x04000000 - P_MOT_SB_OVERCURRENT
Indicates that the motor current has exceeded the continuous current limit specified for the motor. This can occur temporarily during heavy load or fast acceleration conditions and under these circumstances it is normal (motors are normally tolerant of brief current spikes beyond their continuous rating). However, when it occurs over a sustained length of time, it can trigger a P_MOT_SB_OVERLOAD condition.
0x08000000 - P_MOT_SB_BUSCURRENTFAULT
Indicates that excessive current is being drawn from the motor power supply. This condition typically indicates a hard wiring fault that needs to be rectified, for example a phase-to-phase short circuit in a brushless motor.

0x10000000 - P_MOT_SB_POWEROK
Indicates that all the controller power supplies are operating normally.

0x20000000 - P_MOT_SB_ACTIVE
Normally indicates that the controller is executing a motion command.

0x40000000 - P_MOT_SB_ERROR
Indicates an error condition, either listed above or arising as a result of another abnormal condition.

0x80000000 - P_MOT_SB_ENABLED
Indicates that the motor output is enabled and the controller is in charge of maintaining the required position. When the output is disabled, the motor is not controlled by the electronics and can be moved manually, as much as the mechanical construction (such as any leadscrew and gearbox fitted) allows.
This is not full list of all the bits but the remaining bits reflect information about the state of the hardware that in most cases does not affect motion.

See the following table for a list of status bits and applicable controllers.

Motor Controller Status Bits Applicable

Bit	Definition	TDC001	TBD001	KDC101	KBD101	M30X	M30XY	BBD20X	BBD30X
0x0000.0001	P_MOT_SB_CWHARDLIMIT	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0002	P_MOT_SB_CCWHARDLIMIT	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0004	P_MOT_SB_CWSOFTLIMIT	✓	▪	✓	▪	✓	✓	▪	▪
0x0000.0008	P_MOT_SB_CCWSOFTLIMIT	✓	▪	✓	▪	✓	✓	▪	▪
0x0000.0010	P_MOT_SB_INMOTIONCW	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0020	P_MOT_SB_INMOTIONCCW	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0040	P_MOT_SB_JOGGINGCW	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0080	P_MOT_SB_JOGGINGCCW	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0100	P_MOT_SB_CONNECTED	✓	▪	✓	✓	✓	✓	▪	✓
0x0000.0200	P_MOT_SB_HOMING	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0400	P_MOT_SB_HOMED	✓	✓	✓	✓	✓	✓	✓	✓
0x0000.0800	P_MOT_SB_INITILIZING	▪	▪	▪	▪	▪	▪	▪	✓
0x0000.1000	P_MOT_SB_TRACKING	▪	✓	▪	▪	▪	▪	✓	✓
0x0000.2000	P_MOT_SB_SETTLED	▪	✓	▪	▪	▪	▪	✓	✓
0x0000.4000	P_MOT_SB_POSITIONERROR	▪	✓	✓	✓	✓	✓	✓	✓
0x0000.8000	P_MOT_SB_INSTRERROR	▪	✓	▪	▪	▪	▪	✓	▪
0x0001.0000	P_MOT_SB_INTERLOCK	▪	✓	▪	▪	▪	▪	✓	✓
0x0002.0000	P_MOT_SB_OVERTEMP	▪	✓	▪	▪	✓	✓	✓	✓
0x0004.0000	P_MOT_SB_BUSVOLTFAULT	▪	✓	▪	▪	✓	✓	✓	✓
0x0008.0000	P_MOT_SB_COMMUTATIONERROR	▪	✓	▪	▪	▪	▪	✓	✓
0x0010.0000	P_MOT_SB_DIGIP1	✓	✓	✓	✓	✓	✓	✓	▪
0x0020.0000	P_MOT_SB_DIGIP2	▪	▪	✓	✓	✓	✓	▪	▪
0x0040.0000	P_MOT_SB_DIGIP3	▪	▪	▪	▪	▪	▪	▪	▪
0x0080.0000	P_MOT_SB_DIGIP4	▪	▪	▪	▪	▪	▪	▪	▪
0x0100.0000	P_MOT_SB_OVERLOAD	▪	✓	▪	✓	✓	✓	✓	✓
0x0200.0000	P_MOT_SB_ENCODERFAULT	▪	▪	▪	▪	✓	✓	▪	▪
0x0400.0000	P_MOT_SB_OVERCURRENT	▪	▪	▪	▪	✓	✓	▪	✓
0x0800.0000	P_MOT_SB_BUSCURRENTFAULT	▪	▪	▪	▪	✓	✓	▪	✓
0x1000.0000	P_MOT_SB_POWEROK	✓	▪	▪	✓	✓	✓	✓	✓
0x2000.0000	P_MOT_SB_ACTIVE	▪	▪	▪	✓	✓	✓	▪	✓
0x4000.0000	P_MOT_SB_ERROR	✓	✓	✓	▪	✓	✓	✓	✓
0x8000.0000	P_MOT_SB_ENABLED	✓	✓	✓	✓	✓	✓	✓	✓
wMotorCurrent parameter is used and the data is valid		▪	▪	▪	▪	✓	✓	▪	✓

MGMSG_MOT_REQ_USTATUSUPDATE 0x0490

Function: Used to request a status update for the specified motor channel. This request can be used instead of enabling regular updates as described above.

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
90	04	Chan Ident	00	d	s

GET:

See previous details on MGMSG_MOT_GET_USTATUSUPDATE 0x0491.

MGMSG_MOT_ACK_USTATUSUPDATE 0x0492

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called “server alive” must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g., move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
92	04	00	00	d	s

TX 92, 04, 00, 00, 21, 01

MGMSG_ MOT_REQ_STATUSBITS 0x0429
MGMSG_ MOT_GET_STATUSBITS 0x042A

Function: Used to request a "cut down" version of the status update message, only containing the status bits, without data about position and velocity.

SET: N/A

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
29	04	Chan Ident	00	d	s

GET:

Response structure (12 bytes)

6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
2A	04	06	00	d|	s	Chan Ident		Status Bits			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	Word
Status Bits	The status bits are assigned exactly as described in the section detailing the MGMSG_MOT_GET_DCSTATUSUPDATE command.	DWord

MGMSG_ MOT_SUSPEND_ENDOFMOVEMSGS 0x046B

Function: Sent to disable all unsolicited end of move messages and error messages returned by the controller, i.e.

MGMSG_MOT_MOVE_STOPPED
MGMSG_MOT_MOVE_COMPLETED
MGMSG_MOT_MOVE_HOMED

Command structure (6 bytes):

0	1	2	3	4	5
header only					
6B	04	00	00	d	s

MGMSG_ MOT_RESUME_ENDOFMOVEMSGS 0x046C

Function: Sent to resume all unsolicited end of move messages and error messages returned by the controller, i.e.

MGMSG_MOT_MOVE_STOPPED
MGMSG_MOT_MOVE_COMPLETED
MGMSG_MOT_MOVE_HOMED

The command also disables the error messages that the controller sends when an error conditions is detected:

MGMSG_HW_RESPONSE
MGMSG_HW_RICHRESPONSE

This is the default state when the controller is powered up.

Command structure (6 bytes):

0	1	2	3	4	5
header only					
6C	04	00	00	d	s

MGMSG_MOT_SET_TRIGGER 0x0500
MGMSG_MOT_REQ_TRIGGER 0x0501
MGMSG_MOT_GET_TRIGGER 0x0502

Function: This message is used to configure the Motor controller for triggered move operation. It is possible to configure a particular controller to respond to trigger inputs, generate trigger outputs or both respond to and generate a trigger output. When a trigger input is received, the unit can be set to initiate a move (relative, absolute or home). Similarly the unit can be set to generate a trigger output signal when a specified event (e.g move initiated) occurs. For those units configured for both input and output triggering, a move can be initiated via a trigger input while at the same time, a trigger output can be generated to initiate a move on another unit.
The trigger settings can be used to configure multiple units in a master – slave set up, thereby allowing multiple channels of motion to be synchronized. Multiple moves can then be initiated via a single software or hardware trigger command.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
00	05	Chan Ident	Mode	d	s

Note. This message operates differently when used with brushless DC controllers (e.g. BBD20x and TBD001) as opposed to other motor controllers as described in the following paragraphs.

All benchtop stepper controllers (BSC20x,)

field	description	format
Chan Ident	The channel being addressed	char
Mode	This parameter sets the trigger mode and move type to be initiated according to the numerical value entered in bits 0 to 7 as follows Bit 0 (0x01): TRIGIN_ENABLE set to enable physical trigger input Bit 1 (0x02): TRIGOUT_ENABLE set to enable trigger output function (mode set by BIT2 or BIT3 below) Bit 2 (0x04): TRIGOUT_MODEFOLLOW set to enable physical trigger output to mirror trig in Bit 3 (0x08): TRIGOUT_MODEMOVEEND set to enable physical trigger output, remains active (high) until move end Bit 4 (0x10): TRIG_RELMOVE set for relative move on trigger Bit 5 (0x20): TRIG_ABSMOVE set for absolute move on trigger Bit 6 (0x40): TRIG_HOMEMOVE set for home sequence on trigger	char

	Bit 7 (0x80): TRIGOUT_NOTRIGIN set to enable physical trigger output with no physical trigger in (i.e. sw initiated trigger)	

Brushless DC controllers only (BBD20x, BBD30x and TBD001)

field	description	format
Chan Ident	The channel being addressed	char
Mode	This parameter sets the trigger mode and move type according to the numerical value entered in bits 0 to 7 as follows Bit 0 (0x01): TRIGIN_HIGH The Trigger input can be configured to initiate a relative, absolute or homing home, either on the rising or falling edge of the signal driving it. As the trigger input is edge sensitive, it needs to see a logic LOW to HIGH transition ("rising edge") or a logic HIGH to LOW transition ("falling edge") for the move to be started. Additionally, the move parameters must be downloaded to the unit prior to the move using the relevant relative move or absolute move methods as described below. A move already in progress will not be interrupted; therefore external triggering will not work until the previous move has been completed. If this bit is set, the logic state is set HIGH. Bit 1 (0x02): TRIGIN_RELMOVE set to enable trigger in and initiate a relative move (specified using the latest MoveRelative or MoveRelativeEx settings) when a trigger input signal is received. Bit 2 (0x04): TRIGIN_ABSMOVE set to enable trigger in and initiate an absolute move (specified using the latest MoveAbsolute or MoveAbsoluteEx settings) when a trigger input signal is received. Bit 3 (0x08): TRIGIN_HOMEMOVE set to enable trigger in and initiate a home move (specified using the latest MoveHome settings) whan atrigger input signal is received. Bit 4 (0x10): TRIGOUT_HIGH The Trigger output can be configured to be asserted to either logic HIGH or LOW as a function of certain motion-related conditions, such as when a move is in progress (In Motion), complete (Move Complete) or reaches the constant velocity phase on its trajectory (Max Vel). The logic state of the output will remain the same for as long as the chosen condition is true. If this bit is set, the logic state is set HIGH when the following conditions are true. Bit 5 (0x20): TRIGOUT_INMOTION set to enable trigger out (triggered when in motion) Bit 6 (0x40): TRIGOUT_MOTIONCOMPLETE set to enable trigger out (triggered when motion complete) Bit 7 (0x80): TRIGOUT_MAXVELOCITY set to enable trigger out (triggered when axis at maximum velocity)	char

Example: Set the trigger mode for channel 1 of the BBD201 controller as follows:
Trigger Input Rising Edge (High)
Enable trigger input and initiate a Relative Move
Trigger Output Rising Edge (High)
Enable trigger output when move complete.

TX 00, 05, 01, 53, 50, 01

00,05 SET_TRIGGER
01, Channel 1
53, i.e. 01010011
50, destination Generic USB device
01, Source PC

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	05	Chan Ident	00	d	s

Example: Request the trigger mode

TX 01, 05, 01, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
02	05	Chan Ident	Mode	d	s

For structure see SET message above.

MGMSG_MOT_SET_KCUBEMMIPARAMS 0x0520
MGMSG_MOT_REQ_KCUBEMMIPARAMS 0x0521
MGMSG_MOT_GET_KCUBEMMIPARAMS 0x0522

This message is applicable only to KST101, KDC101, KBD101 and BBD30x units

Function: This message is used to configure the operating parameters of the top panel wheel (Joystick).

SET
Command structure (42 bytes)
6 byte header followed by 36 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
20	05	1C	00	d	s	Chan Ident		JSMode		JSMaxVel	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JSMaxVel		JSAccn				DirSense		PreSetPos1			

24	25	26	27	28	29	30	31	32	33
Data									
PreSetPos2				DispBrightness		DispTimeout		DispDimLevel	

34	35	36	37	38	39	40	41
Data							
PreSetPos3				JSSensitivity		Reserved	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
JSMode	This parameter specifies the operating mode of the wheel/joy stick as follows: 1 Velocity Control Mode - Deflecting the wheel starts a move with the velocity proportional to the deflection. The maximum velocity (i.e. velocity corresponding to the full deflection of the joystick wheel) and acceleration are specified in the MaxVel and MaxAccn parameters. 2 Jog Mode - Deflecting the wheel initiates a jog move, using the parameters specified by the SetJogStepSize and SetJogVelParams methods. Keeping the wheel deflected repeats the move automatically after the current move has completed. 3 Go To Position Mode - Deflecting the wheel starts a move from the current position to one of the two predefined "teach" positions. The teach positions are specified in number of steps from the home position in the PresetPos1 and PresetPos2 parameters.	word
JSMaxVel	The max velocity of a move initiated by the top panel	long

	velocity wheel.	
JSAccn	The max acceleration of a move initiated by the top panel velocity wheel	long
DirSense	This parameter specifies the direction of a move initiated by the velocity wheel as follows: 0 Wheel initiated moves are disabled. Wheel used for menuing only. 1 Upwards rotation of the wheel results in a positive motion (i.e. increased position count). The following option applies only when the JSMode is set to Velocity Control Mode (1). If set to Jog Mode (2) or Go to Position Mode (3), the following option is ignored. 2 Upwards rotation of the wheel results in a negative motion (i.e. decreased position count).	word
PresetPos1	The preset position 1 when operating in go to position mode, measured in position steps from the home position.	long
PresetPos2	The preset position 2 when operating in go to position mode, measured in position steps from the home position.	long
DispBrightness	In certain applications, it may be necessary to adjust the brightness of the LED display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again.	word
DispTimeout	'Burn In' of the display can occur if it remains static for a long time. To prevent this, the display is automatically dimmed after the time interval specified in the DispTimeout parameter has elapsed. Set in minutes in the range 0 (never dimmed) to 480. The dim level is set in the DispDimLevel parameter below.	word
DispDimLevel	The dim level, as a value from 0 (Off) to 10 (brightest) but is also limited by the DispBrightness parameter.	word
PresetPos3	**Applicable to BBD30x Only.** The preset position 3 when operating in go to position mode, measured in position steps from the home position.	long
wJSSensitivity	**Applicable to BBD30x Only.** Joystick sensitivity 0 to 65535 representing zero to maximum sensitivity	word
wReserved		word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
21	05	Chan Ident	00	d	s

Example: Request the settings for the top panel wheel

TX 21, 05, 01, 00, 50, 01

GET:

Response structure (6 bytes):

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
22	05	1C	00	d	s	Chan Ident		JSMode		JSMaxVel	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JSMaxVel		JSAccn				DirSense		PreSetPos1			

24	25	26	27	28	29	30	31	32	33
Data									
PreSetPos2				DispBrightness		DispTimeout		DispDimLevel	

34	35	36	37	38	39	40	41
Data							
PreSetPos3				JSSensitivity		Reserved	

For structure see SET message above.

MGMSG_MOT_SET_KCUBETRIGIOCONFIG 0x0523
MGMSG_MOT_REQ_KCUBETRIGIOCONFIG 0x0524
MGMSG_MOT_GET_KCUBETRIGIOCONFIG 0x0525

This message is applicable only to KST101, KDC101 and KBD101 units

Function: The K-Cube motor controllers have two bidirectional trigger ports (TRIG1 and TRIG2) that can be used to read an external logic signal or output a logic level to control external equipment. Either of them can be independently configured as an input or an output and the active logic state can be selected High or Low to suit the requirements of the application. Electrically the ports output 5 Volt logic signals and are designed to be driven from a 5 Volt logic.
When the port is used in the input mode, the logic levels are TTL compatible, i.e. a voltage level less than 0.8 Volt will be recognised as a logic LOW and a level greater than 2.4 Volt as a logic HIGH. The input contains a weak pull-up, so the state of the input with nothing connected will default to a logic HIGH. The weak pull-up feature allows a passive device, such as a mechanical switch to be connected directly to the input.
When the port is used as an output it provides a push-pull drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuity.

Warning: do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

SET

Command structure (28 bytes)

6 byte header followed by 22 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
23	05	0C	00	d	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18 to 28
Data						
Trig2Mode		Trig2Polarity		Reserved		Reserved

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
Trig1Mode	TRIG1 operating mode	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) l.	word
Trig2Mode	TRIG2 operating mode	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low)	word
Reserved	Bytes 16 to 28	word

Input Trigger Modes

When configured as an input, the TRIG ports can be used as a general purpose digital input, or for triggering a relative, absolute or home move as follows:

0x00 The trigger IO is disabled
0x01 General purpose logic input (read through status bits using the MOT_GET_STATUSBITS message).
0x02 Input trigger for relative move.
0x03 Input trigger for absolute move.
0x04 Input trigger for home move.

When used for triggering a move, the port is edge sensitive. In other words, it has to see a transition from the inactive to the active logic state (Low->High or High->Low) for the trigger input to be recognized. For the same reason a sustained logic level will not trigger repeated moves. The trigger input has to return to its inactive state first in order to start the next trigger.

Output Trigger Modes

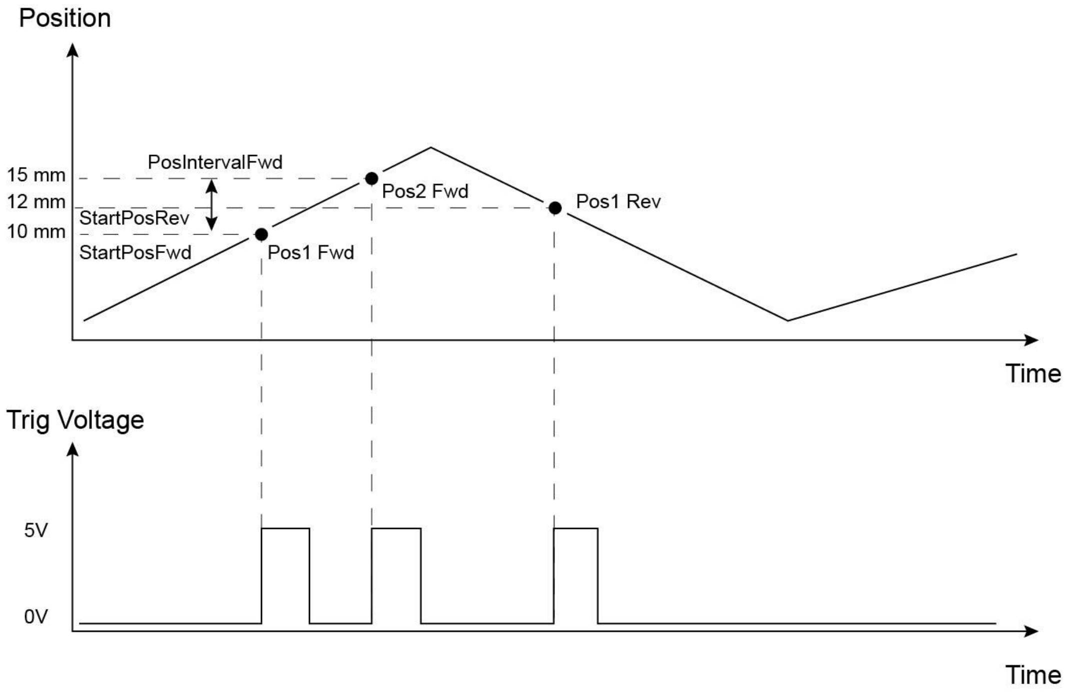
When configured as an output, the TRIG ports can be used as a general purpose digital output, or to indicate motion status or to produce a trigger pulse at configurable positions as follows:

0x0A General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).
0x0B Trigger output active (level) when motor 'in motion'. The output trigger goes high (5V) or low (0V) (as set in the lTrig1Polarity and lTrig2Polarity parameters) when the stage is in motion.
0x0C Trigger output active (level) when motor at 'max velocity'.
0x0D Trigger output active (pulsed) at pre-defined positions moving forward (set using StartPosFwd, IntervalFwd, NumPulsesFwd and PulseWidth parameters in the SetKCubePosTrigParams message). Only one Trigger port at a time can be set to this mode.
0x0E Trigger output active (pulsed) at pre-defined positions moving backwards (set using StartPosRev, IntervalRev, NumPulsesRev and PulseWidth parameters in the SetKCubePosTrigParams message). Only one Trigger port at a time can be set to this mode.
0x0F Trigger output active (pulsed) at pre-defined positions moving forwards and backward. Only one Trigger port at a time can be set to this mode.

Trigger Out Position Steps

In the last three modes described above, the controller outputs a configurable number of pulses, of configurable width, when the actual position of the stage matches the position values configured as the Start Position and Position Interval - see SetKCubePosTrigParams message. These modes allow external equipment to be triggered at exact position values. The position pulses are generated by dedicated hardware, allowing a very low latency of less than 1 usec. The low latency of this triggering mode provides a very precise indication of a position match (assuming a stage velocity of 10 mm/sec, the less than 1 usec latency would in itself only result in a 10 nm position uncertainty, which is normally well below the accuracy limitations of the mechanics.)

Using the last three modes above, position triggering can be configured to be unidirectional (forward or reverse only) or bidirectional (both). In bidirectional mode the forward and reverse pulse sequences can be configured separately. A cycle count setting (set in the SetKCubePosTrigParams message, lNumCycles parameter) allows the uni- or bidirectional position triggering sequence to be repeated a number of times.



Example for a move from 0 to 20 mm and back.

In forward direction: The first trigger pulse occurs at 10 mm (StartPosFwd), the next trigger pulse occurs after another 5 mm (PosIntervalFwd), the stage then moves to 20 mm.

In reverse direction: The next trigger occurs when the stage gets to 12 mm.

Please note that position triggering can only be used on one TRIG port at a time, as there is only one set of position trigger parameters.

The operation of the position triggering mode is described in more detail in the SetKCubePosTriggerParams method.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
24	05	Chan Ident	00	d	s

Example: Request the settings for the top panel wheel

TX 24, 05, 01, 00, 50, 01

GET:

Response structure (18 bytes):
6 byte header followed by 12 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
25	05	0C	00	d	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17
Data					
Trig2Mode		Trig2Polarity		Reserved	

For structure see SET message above.

MGMSG_MOT_SET_KCUBEPOSTRIGPARAMS 0x0526
MGMSG_MOT_REQ_KCUBEPOSTRIGPARAMS 0x0527
MGMSG_MOT_GET_KCUBEPOSTRIGPARAMS 0x0528

This message is applicable only to KST101, KDC101 and KBD101 units

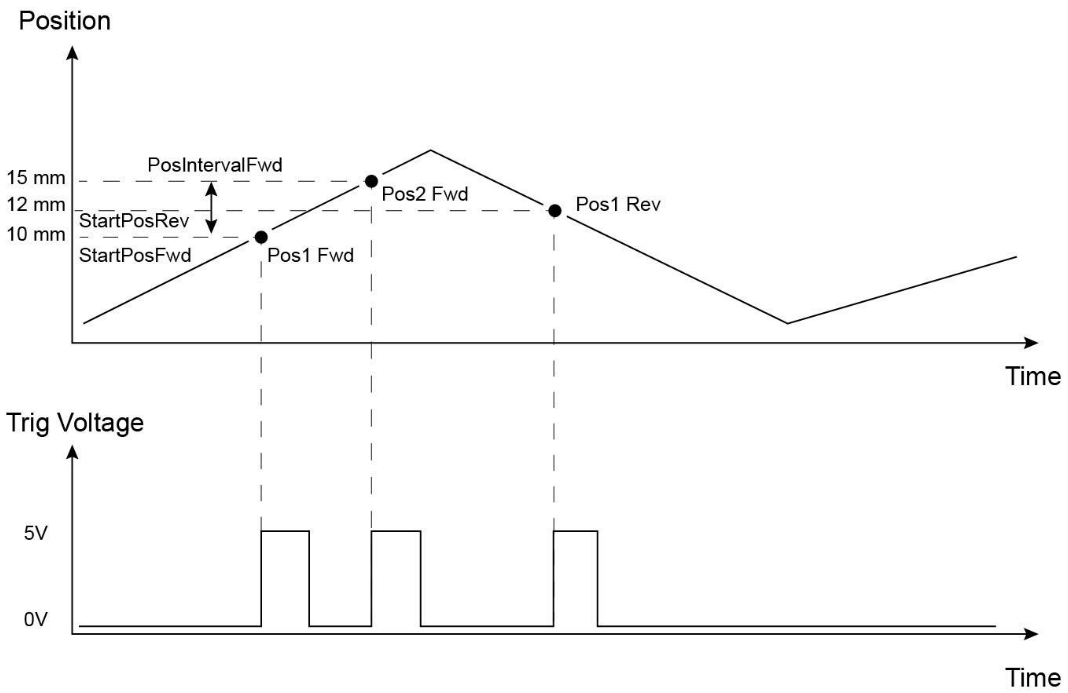
Function: The K-Cube motor controllers have two bidirectional trigger ports (TRIG1 and TRIG2) that can be set to be used as input or output triggers. This method sets operating parameters used when the triggering mode is set to a trigger out position steps mode by calling the SetKCubeTrigIOConfig message.

As soon as position triggering is selected on either of the TRIG ports, the port will assert the inactive logic state. As the stage moves in its travel range and the actual position matches the position set in the StartPosFwd parameter, the TRIG port will output its active logic state. The active state will be output for the length of time specified by the PulseWidth parameter, then return to its inactive state and schedule the next position trigger point at the "StartPosFwd value plus the value set in the fPosIntervalFwd parameter. Thus when this second position is reached, the TRIG output will be asserted to its active state again. The sequence is repeated the number of times set in the NumPulsesFwd parameter.

When the number of pulses set in the NumPulsesFwd parameter has been generated, the trigger engine will schedule the next position to occur at the position specified in the StartPosRev parameter. The same sequence as the forward direction is now repeated in reverse, except that the PosIntervalRev and NumPulsesRev parameters apply. When the number of pulses has been output, the entire forward-reverse sequence will repeat the number of times specified by NumCycles parameter. This means that the total number of pulses output will be NumCycles x (NumPulsesFwd + NumPulsesRev).

Once the total number of output pulses have been generated, the trigger output will remain inactive.

When a unidirectional sequence is selected, only the forward or reverse part of the sequence will be activated.



Example for a move from 0 to 20 mm and back.

In forward direction: The first trigger pulse occurs at 10 mm (StartPosFwd), the next trigger pulse occurs after another 5 mm (PosIntervalFwd), the stage then moves to 20 mm.

In reverse direction: The next trigger occurs when the stage gets to 12 mm.
Note that the position triggering scheme works on the principle of always triggering at the next scheduled position only, regardless of the actual direction of movement. If, for example, a position trigger sequence is set up with the forward start position at 10 mm, but initially the stage is at 15 mm, the first forward position trigger will occur when the stage is moving in the reverse direction. Likewise, if the stage does not complete all the forward position trigger points, the reverse triggering will not activate at all. For normal operation it is assumed that all trigger points will be reached during the course of the movement.

SET

Command structure (40 bytes)

6 byte header followed by 34 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
26	05	22	00	d	s	Chan Ident		StartPosFwd			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
IntervalRev				NumPulsesRev				PulseWidth			

36	37	38	39
Data			
NumCycles			

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
StartPosFwd -	When moving forward, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalFwd	When moving forward, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesFwd	Number of output pulses during a forward move.	long
StartPosRev -	When moving backwards, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalRev	When moving backwards, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesRev	Number of output pulses during a backwards move.	long
PulseWidth	Trigger output pulse width (from 1 µs to 1000000 µs).	long
NumCycles	Number of forward/reverse move cycles.	long

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
27	05	Chan Ident	00	d	s

Example: Request the settings for the top panel wheel

TX 27, 05, 01, 00, 50, 01

GET:

Response structure (40 bytes):

6 byte header followed by 34 byte data packet.

0	1	2	3	4	5	6	7		8	9	10	11
header							*Data*					
28	05	22	00	d	s	Chan Ident			StartPosFwd			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
IntervalRev				NumPulsesRev				PulseWidth			

36	37	38	39
Data			
IntervalFwd			

For structure see SET message above.

MGMSG_MOT_SET_KCUBEKSTLOOPPARAMS 0x0529
MGMSG_MOT_REQ_KCUBEKSTLOOPPARAMS 0x052A
MGMSG_MOT_GET_KCUBEKSTLOOPPARAMS 0x052B

This message is applicable only to KST101 and BSC20X units

Function: Used to set the position control loop parameters for the specified motor channel.
The motion processor within the controller uses a position control loop to determine the motor command output. The purpose of the position loop is to match the actual motor position and the demanded position. This is achieved by comparing the demanded position with the actual position to create a position error, which is then passed through a digital PID-type filter. The filtered value is the motor command output.

SET:
Command structure (42 bytes)
6 byte header followed by 36 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
29	05	24	00	d|	s	Chan Ident		LoopMode		Prop			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Int				Derivative				PIDClip			

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Data															
PIDTol				EncoderConst				Not Used							

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
LoopMode	Sets Open or Closed Loop as follows: 1 Open Loop 2 Closed Loop	word
Prop	The proportional gain. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 16777216.	long
Int	The integral gain. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 16777216.	long
Derivative	The derivative gain. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 16777216.	long
PIDClip	The PIDClip parameter is used to cap the value of the PID loop to prevent runaway at the output. It accepts values in the range 0 to 16777216. If set to 0 then the output of the PID loop is ignored.	long
PIDTol	Value below which the output of PID generator is effectively	long

	deemed to be zero to avoid continual cycle about set point	
EncoderConst	This is a conversion factor from Encoder counts to microsteps. If set to 0, then no encoder is fitted to the stage.	DWord

Example: Set the PID parameters as follows:
Loop Mode: Closed Loop
Prop: 20000
Int: 1000
Derivative: 100
PIDClip: 100,000
PidTol: 200
EncoderConst: 4292282941 (see note below

TX 29, 05, 24, 00, D0, 01, 01, 00, 02, 00, 20, 4E, 00, 00, E8, 03, 00, 00, 64, 00, 00, 00, 00, E1, F5, 05, C8, 00, 00, 00, C3, F5. 28, 00, 00, 00, 00, 00, 00, 00, 00, 00,

Header: 29, 05, 24, 00, D0, 01: Set_KCubeKSTLoopParams, 36 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1 (always set to 1 for BSC201)
LoopMode: 02, 00 : Closed Loop
Prop: *20, 4E, 00, 00:* Set the proportional term to 20000
Int: *E8, 03,:* Set the integral term to 1000
Derivative: 64, 00,: Set the derivative term to 100
PIDClip: 00, E1, F5, 05,: Set the integral limit to 100,000,000
PIDTol: C8, 00, 00, 00
EncoderConstl: C3, F5, 28, 00, : Set the Enccoder Constant to 4292282941.

Note. Calculating the EncoderConst Value
Each stage has a specific constant for converting encoder counts to microsteps. For the LNR50SE stage, this value is 4292282941.

For example
Encoder resolution = 100 nm
Stepper resolution = 409600 microsteps/turn/mm
= 2.44 nm per step
Therefore no. of µsteps per encoder count = 100 nm/2.44 = 40.96.

The chip inside the controller uses 16.16 bit format, where 16 bits represent the integer and 16 bit are for the fraction.

Interger part 40 = Hex28 = 0X0028
Fraction part 0.96/1/65536 = 62914.56 = F5C3
Therefore EncoderConst value = **0028F5C3**

For negative values, we must find the 2s compliment value…
28F5C3 = 0000 0000 0010 1000.1111 0101 1100 0011
2s comp = 1111 1111 1101 0111.0000 1010 0011 1100 + 1
= **FFD7.0A3D**

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
2A	05	Chan Ident	00	d	s

GET:
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
2B	05	24	00	d|	s	Chan Ident		LoopMode		Prop			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Int				Derivative				PIDClip			

26	27	28	29	30	31	32	33	34	35
Data									
PIDTol				EncoderConst				Not Used	

For structure see Set message above.

MGMSG_MOT_SET_KCUBEPOSTRIGPARAMS 0x0526
MGMSG_MOT_REQ_KCUBEPOSTRIGPARAMS 0x0527
MGMSG_MOT_GET_KCUBEPOSTRIGPARAMS 0x0528

This message is applicable only to KST101, KDC101 and KBD101 units

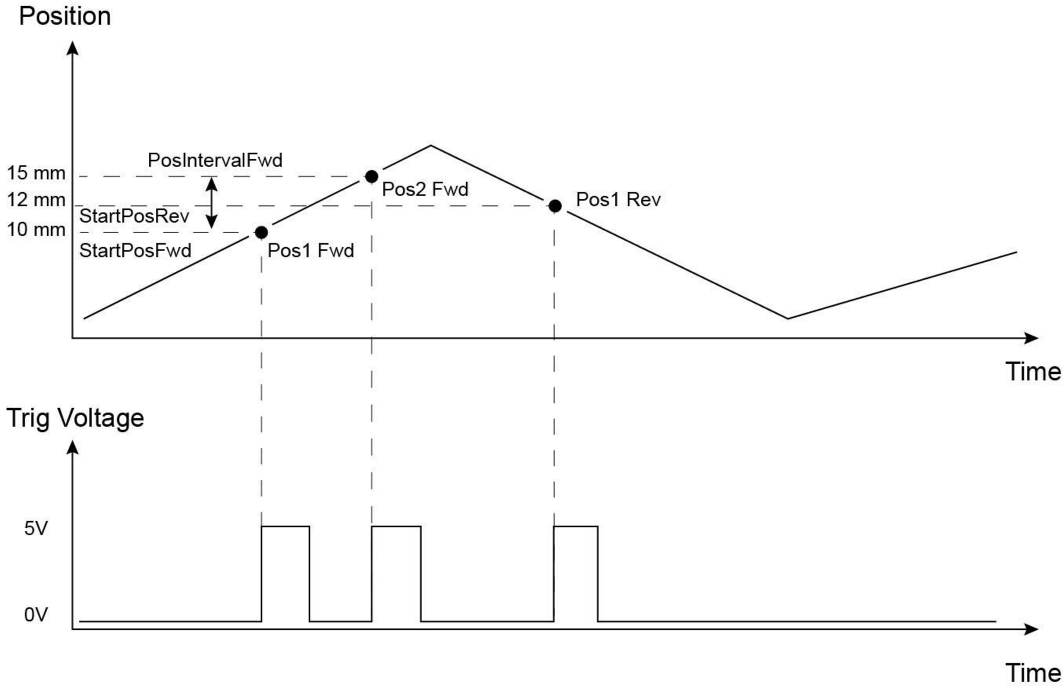
Function: The K-Cube motor controllers have two bidirectional trigger ports (TRIG1 and TRIG2) that can be set to be used as input or output triggers. This method sets operating parameters used when the triggering mode is set to a trigger out position steps mode by calling the SetKCubeTrigIOConfig message.

As soon as position triggering is selected on either of the TRIG ports, the port will assert the inactive logic state. As the stage moves in its travel range and the actual position matches the position set in the StartPosFwd parameter, the TRIG port will output its active logic state. The active state will be output for the length of time specified by the PulseWidth parameter, then return to its inactive state and schedule the next position trigger point at the "StartPosFwd value plus the value set in the fPosIntervalFwd parameter. Thus when this second position is reached, the TRIG output will be asserted to its active state again. The sequence is repeated the number of times set in the NumPulsesFwd parameter.

When the number of pulses set in the NumPulsesFwd parameter has been generated, the trigger engine will schedule the next position to occur at the position specified in the StartPosRev parameter. The same sequence as the forward direction is now repeated in reverse, except that the PosIntervalRev and NumPulsesRev parameters apply. When the number of pulses has been output, the entire forward-reverse sequence will repeat the number of times specified by NumCycles parameter. This means that the total number of pulses output will be NumCycles x (NumPulsesFwd + NumPulsesRev).

Once the total number of output pulses have been generated, the trigger output will remain inactive.

When a unidirectional sequence is selected, only the forward or reverse part of the sequence will be activated.



Example for a move from 0 to 20 mm and back.

In forward direction: The first trigger pulse occurs at 10 mm (StartPosFwd), the next trigger pulse occurs after another 5 mm (PosIntervalFwd), the stage then moves to 20 mm.

In reverse direction: The next trigger occurs when the stage gets to 12 mm.
Note that the position triggering scheme works on the principle of always triggering at the next scheduled position only, regardless of the actual direction of movement. If, for example, a position trigger sequence is set up with the forward start position at 10 mm, but initially the stage is at 15 mm, the first forward position trigger will occur when the stage is moving in the reverse direction. Likewise, if the stage does not complete all the forward position trigger points, the reverse triggering will not activate at all. For normal operation it is assumed that all trigger points will be reached during the course of the movement.

SET

Command structure (40 bytes)

6 byte header followed by 34 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
26	05	22	00	d	s	Chan Ident		StartPosFwd			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
IntervalRev				NumPulsesRev				PulseWidth			

36	37	38	39
Data			
NumCycles			

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
StartPosFwd -	When moving forward, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalFwd	When moving forward, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesFwd	Number of output pulses during a forward move.	long
StartPosRev -	When moving backwards, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalRev	When moving backwards, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesRev	Number of output pulses during a backwards move.	long
PulseWidth	Trigger output pulse width (from 1 µs to 1000000 µs).	long
NumCycles	Number of forward/reverse move cycles.	long

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
27	05	Chan Ident	00	d	s

Example: Request the settings for the top panel wheel

TX 27, 05, 01, 00, 50, 01

GET:

Response structure (40 bytes):

6 byte header followed by 34 byte data packet.

0	1	2	3	4	5	6	7		8	9	10	11
header							*Data*					
28	05	22	00	d	s	Chan Ident			StartPosFwd			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
IntervalRev				NumPulsesRev				PulseWidth			

36	37	38	39
Data			
IntervalFwd			

For structure see SET message above.

MGMSG_MOT_SET_MOTTRIGIOCONFIG 0x0260
MGMSG_MOT_REQ_MOTTRIGIOCONFIG 0x0261
MGMSG_MOT_GET_MOTTRIGIOCONFIG 0x0262

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: The BBD30x brushless motor controllers offer additional triggering functions that expand the triggering capabilities of the legacy BBD20X controllers. In the legacy controller family, triggering is
(a) implemented on a per-channel basis, i.e. each motor channel had its own dedicated trigger input and output.
(b) for the inputs, hardware triggering can be used to trigger an absolute, relative and home moves.
(c) the outputs can be asserted when the stage is in motion or reaches its maximum velocity.

The BBD30x controllers expand this functionality and add the following features:
(a) For ease of connectivity, the main trigger features are brought out on BNC connectors. This allows the use of off-the-shelf BNC cables for connecting to external equipment rather than having to interface to D-type connectors.
(b) The trigger inputs and outputs are not hard wired to a particular I/O connector. In input mode, the same connector can be used to trigger multiple motor channels. In output mode, any I/O connector can be used to output a trigger signal from a motor channel.
(c) Low latency position triggering has been added. This operates at the speed of the hardware signals involved, offering delays measured in tens of nanoseconds.
(d) An analogue monitor feature has been added, allowing various system variables to be monitored using an external oscilloscope.

A block diagram of the BBD30x BNC trigger I/O is shown below:

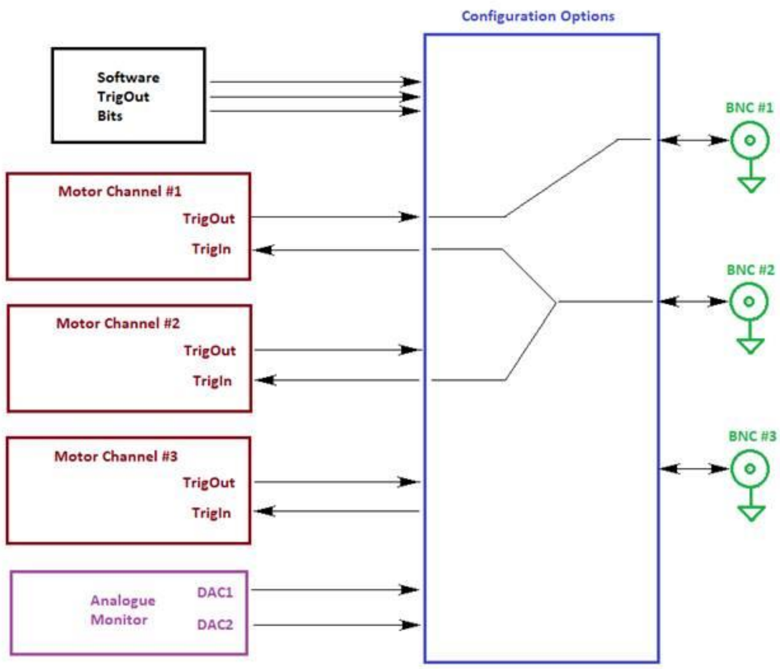


Figure 1 BBD30x BNC trigger I/O Schematic

Main features:

- Each BNC port can be configured to be a digital input, a digital output or an analogue monitor output (only on BNC #1 and BNC #2).
- Each motor channel also has a trigger input and a trigger output signal.
- When a BNC I/O port is configured as an input, its signal can be routed to any one or more motor channels. This, for example, as shown above, BNC #2 is routed to both Motor Channel #1 and Motor Channel #2.
- When configured as an output, the BNC I/O port can be driven by any of the motor channels (but only one, in order to avoid the possibility of conflicts between different motor channels driving the same output).
- When configured as an analogue monitor, the BNC I/O port can be used to output an analogue voltage in the range of 0 to +5V. This can be assigned to a number of system variables and provides a very low latency way of monitoring the state of the system as the signal follows any changes to the system variable at the speed of the hardware. For example, absolute position can be monitored to aid PID tuning.
- The BBD301 and BBD302 controllers provide two BNC trigger I/O ports whereas the BBD303 provides three.

Warning: do not drive the TRIG ports from any voltage source that can produce an output more than the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

SET

Command structure (58 bytes)

6 byte header followed by 52 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
60	02	34	00	d	s	Channel		TrigInMode		TrigInPolarity		TrigInSource	

14	15	16	17	18	19	20	21	22	23	24	25
Data											
TrigOutMode		TrigOutPolarity		StartPosFwd				IntervalFwd			

26	27	28	29	30	31	32	33	34	45	36	37
Data											
NumPulsesFwd				StartPosRev				IntervalRev			

38	39	40	41	42	43	44	45	46	47	48	49	50	51
Data													
NumPulsesRev				PulseWidth				NumCycles				Reserved	

50	51	52	53	54	55	56	57
Data							
Reserved		Reserved		Reserved		Reserved	

Data Structure:

field	description	format
Channel	The channel being addressed (0x01, 0x02 or 0x03) encoded as a 16-bit word (e.g., 0x01 0x00)	word

TrigInMode	The trigger input operating mode (see Input Trigger Modes below)	word
TrigInPolarity	The active state of input trigger (i.e. logic high or logic low). HIGH 0x01 LOW 0x02	word
TrigInSource	The trigger input source SOFTWARE 0x00 Not Used BNC1 0x01 The trigger input source is BNC #1 BNC2 0x02 The trigger input source is BNC #2 BNC3 0x03 The trigger input source is BNC #3	word
TrigOutMode	The trigger output operating mode (see Output Trigger Modes below)	word
TrigOutPolarity	The active state of output trigger (i.e. logic high or logic low). HIGH 0x01 LOW 0x02	word
StartPosFwd	When moving forward, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalFwd	When moving forward, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesFwd	Number of output pulses during a forward move.	long
StartPosRev	When moving backwards, this is the stage position [in position counts - encoder counts or microsteps] to start the triggering sequence.	long
IntervalRev	When moving backwards, this is the interval [in position counts - encoder counts or microsteps] at which to output the trigger pulses.	long
NumPulsesRev	Number of output pulses during a backwards move.	long
PulseWidth	Trigger output pulse width (from 1 µs to 1000000 µs).	long
NumCycles	Number of forward/reverse move cycles.	long
Reserved		word
Reserved		word
Reserved		word
Reserved		word

Input Trigger Modes

The controller has up to 3 BNC connectors on the rear panel and each of these can be configured as a Trigger Input or a trigger output. This is set in the *Mode* parameter of the SET_IOCONFIG message.

When set to an input, the input mode is set in the *TrigInMode* parameter and can be set to home the stage, move the stage by a relative amount, or move the stage to a specified absolute position.

The source for the input signal is set in the *TrigInSource* parameter. Note that Input Source – SOFTWARE(0x00) is not used at this time.

The rising or falling edge of a Trigger In signal initiates the action and is set in the *TrigInPolarity* parameter. The rising edge refers to a transition from logic LOW to HIGH, and the falling edge refers to a transition from logic HIGH to LOW. Since a move already in progress will not be interrupted, the stage will not respond to an external Trigger In signal if in the process of executing a move.

The trigger settings can be used to specify whether Trigger In is disabled or will respond to the rising or falling edge of a Trigger In signal. When an absolute move is selected, the target position is specified in the *SET_MOVEABSPARAMS* message *Absolute Position* parameter. If the signal is specified to initiate a relative move, both the direction of the move and the relative distance can be specified in the

SET_MOVERELPARAMS message *Relative Distance* parameter. In order to avoid unexpended moves being executed on start-up, the trigger input settings are not retained in memory and will default to the input being disabled on power-up. Also be aware that transients generated when powering off a function generator connected to the Trigger In port can also be interpreted as a Trigger In signal.

Input Trigger Mode options are set as follows:

0x00 *Disabled* – triggering operation is disabled

0x01 *GPI* – General purpose input

0x02 *Relative Move* – a relative move is initiated on the selected channel when an input signal is received. The Input Source (BNC1, BNC2 etc.), the Polarity (High or Low) of the trigger signal, the relative distance to move and the direction of travel are specified in their associated parameter fields.

0x03 *Absolute Move* – an absolute move is initiated on the selected channel when an input signal is received. The Input Source (BNC1, BNC2 etc.), the Polarity (High or Low) of the trigger signal and the absolute distance to move are specified in their associated parameter fields.

0x04 *Home Move* – a home move is initiated on the selected channel when an input signal is received. The Input Source (BNC1, BNC2 etc.), and the Polarity (High or Low) of the trigger signal are specified in their associated parameter fields.

0x05 *Stop* - a stop command is initiated on the selected channel when an input signal is received. The Input Source (BNC1, BNC2 etc.), and the Polarity (High or Low) of the trigger signal are specified in their associated parameter fields.

Output Trigger Modes

The controller has up to 3 BNC connectors on the rear panel and each of these can be configured as a trigger input or a trigger output. This is set in the *Mode* parameter of the SET_IOCONFIG message.

When set to an output, the output mode is set in the *TrigOutMode* parameter and can be set to any of the options described below.

The motor channel's output trigger signal can be routed to any of the BNC connectors on the rear panel (set in the SET_IOCONFIG message). This output signal is either logic High or Low as set in the *TrigOutPolarity* parameter (LOW by default).

The Trigger Out output settings can be retained in memory and the settings will be automatically applied once phase initialization has completed after the next power-up. Whilst this can be advantageous in some applications, please note that immediately after power-up while the unit is going through its normal boot-up and initialization process, the Trigger Out output may not be in its expected state.

Output Trigger options are set as follows:

0x0a *GPO* – General purpose logic output. The output is

0x0B *In Motion* - Trigger output active (level) when motor 'in motion'. The output trigger goes high (5V) or low (0V) (as set in the Trig 1. Polarity and Trig. 2 Polarity parameters) when the stage is in motion.

0x0C *At Max Velocity* - Trigger output active (level) when motor at 'max velocity'.

0x0D *At Position Steps Fwd* - Trigger output active (pulsed) at pre-defined positions moving forward. Only one Trigger port at a time can be set to this mode. See Trigger Out Position Steps below for further details.

0x0E *At Position Steps Rev* - Trigger output active (pulsed) at pre-defined positions moving backwards. Only one Trigger port at a time can be set to this mode. See Trigger Out Position Steps below for further details.

0x0F *At Position Steps Both* - Trigger output active (pulsed) at pre-defined positions moving forwards and backward. Only one Trigger port at a time can be set to this mode. See Trigger Out Position Steps below for further details.

0x10 *At Forward Limit* – Trigger output active (level) when the forward limit switch is made.

0x11 *At Reverse Limit* – Trigger output active (level) when the reverse limit switch is made.

0x12 *At Either Limit* – Trigger output active (level) when either the forward or the reverse limit switch is activated

Trigger Out Position Steps

Note

If the trigger mode is not set to one of the three position modes described previously, then the following parameters are not applicable.

As soon as a position triggering mode is selected on any of the BNC ports, the port will assert the inactive logic state, set in the *TrigOutPolarity* parameter.

As the stage moves in its travel range and the actual position matches the position set in the *StartPosFwd* parameter, the BNC port will output its active logic state. The active state will be output for the length of time specified by the Trigger *PulseWidth* parameter, then return to its inactive state and schedule the next position trigger point at the *StartPosFwd* value plus the value set in *IntervalFwd* parameter.

Thus when this second position is reached, the BNC output will be asserted to its active state again. The sequence is repeated the number of times set in the *NumPulsesFwd* parameter.

When the number of pulses set in the *NumPulsesFwd* parameter has been generated, the trigger engine will schedule the next position to occur at the position specified in the *StartPosRev* parameter. The same sequence as the forward direction is now repeated in reverse, except that the Reverse setting parameters apply. When the number of pulses has been output, the entire forward-reverse sequence will repeat the number of times specified by *NumCycles* parameter. This means that the total number of pulses output will be *NumCycles x (NumPulsesFwd* + *NumPulsesRev)*.

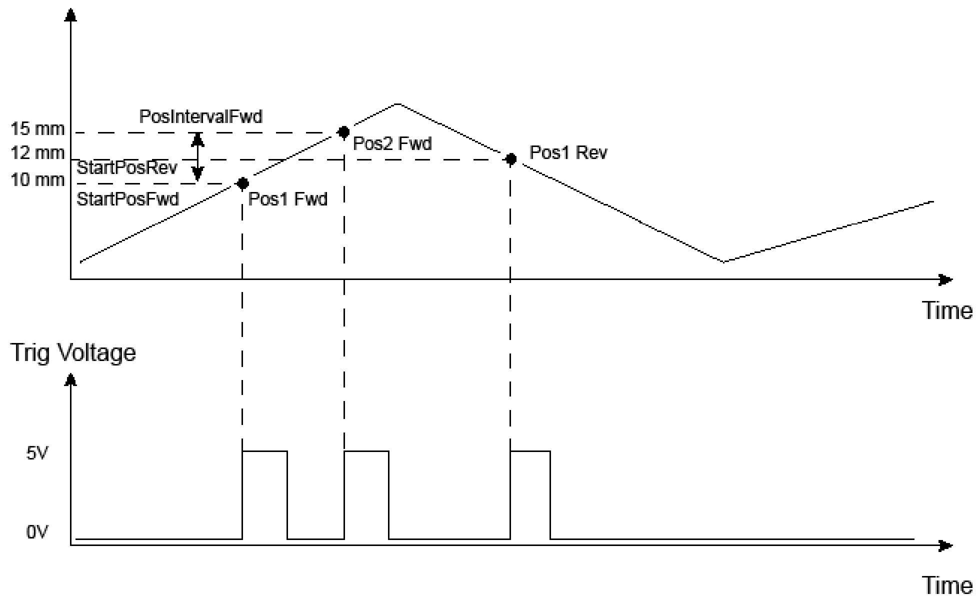


Figure 1 Position Steps Triggering

Example for a move from 0 to 20 mm and back.

In forward direction: The first trigger pulse occurs at 10 mm (StartPosFwd), the next trigger pulse occurs after another 5 mm (PosIntervalFwd), the stage then moves to 20 mm.

In reverse direction: The next trigger occurs when the stage gets to 12 mm.

Trigger Polarity

The polarity of the trigger pulse is specified in the *TrigInPolarity* and *TrigOutPolarity* parameters as follows:

Trigger High - The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).

Trigger Low - The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

Triggering Latency

The detection of whether a trigger condition has occurred is carried out periodically at 102 µs intervals. As a result, there is a maximum 102 µs delay between the condition occurring and the trigger output being updated. The following timing diagram illustrates this latency.

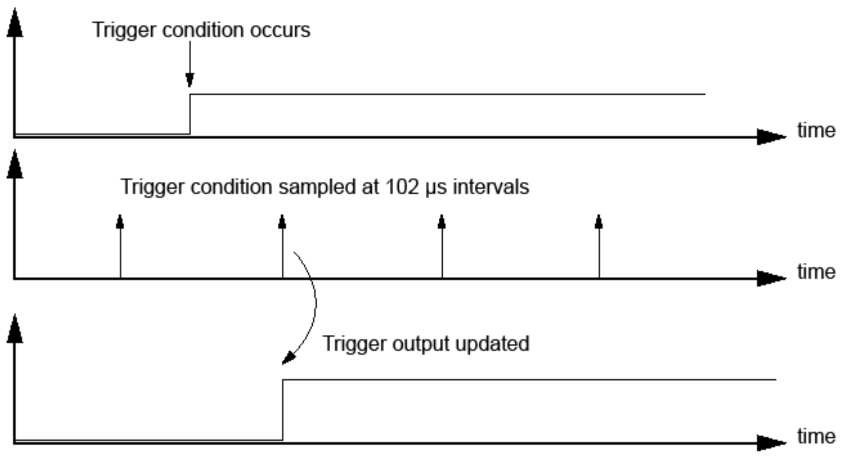


Figure 2 Triggering Latency

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
61	02	Chan Ident	00	d	s

Example: Request the settings for the Trigger IO

TX 61, 02, 01, 00, 50, 01

GET:
Command structure (58 bytes)
6 byte header followed by 52 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_IOCONFIG 0x0263
MGMSG_MOT_REQ_IOCONFIG 0x0264
MGMSG_MOT_GET_IOCONFIG 0x0265

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: This command is used to configure the BNC connectors on the rear panel of the BBD30x units. The connectors can be associated with a specific motor channel, and be set to an input or an output as follows.

SET
Command structure (12 bytes)
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
63	02	0C	00	d	s	IOPort		Mode		OutSource	

Data Structure:

field	description	format
IOPort	The rear panel BNC Connector number being configured.	word
Mode	The operating mode of the connector: DIGIIN 0x00 The I/O port is configured as a digital input. DIGIOUT 0x01 The I/O port is configured as a digital output ANALOGOUT 0x02 The I/O port is configured as an analog output. It port can be used to output an analogue voltage in the range of 0 to +5V. This can be assigned to a number of system variables and provides a very low latency way of monitoring the state of the system as the signal follows any changes to the system variable at the speed of the hardware. For example, absolute position can be monitored to aid PID tuning. If this option is selected, the output is configured using the SET_ANALOGMONITORCONFIG (0x0269) message. Only BNC #1 or BNC#2 connectors can be configured as Analog outputs.	word
OutSource	The destination for the output signal. SOFTWARE 0x00 The state of the output is software defined MOTCHAN1 0x01 The state of the output is Motor Channel #1 MOTCHAN2 0x02 The state of the output is Motor Channel #2 MOTCHAN3 0x03 The state of the output is Motor Channel #3	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
64	02	Chan Ident	03	d	s

Byte 3 of the request message specifies the port number being addressed, e.g. BNC #3 = 0x03.

Example: Request the settings for IO Port BNC 3

TX 64, 02, 01, 03, 50, 01

GET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_AUXIOCONFIG 0x0266
MGMSG_MOT_REQ_AUXIOCONFIG 0x0267
MGMSG_MOT_GET_AUXIOCONFIG 0x0268

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: In addition to the functionality provided by the BNC connectors, the AUX I/O connector (37-way D-type) on the rear panel provides further flexible options for connecting external digital I/O signals.
The connector provides

- 4 single ended input ports
- 4 single ended output ports
- 2 differential input ports
- One RS-232 port (Rx and Tx)
- 12 differential output ports

The 12 differential output ports offer the user the choice to expose a buffered version of the 3 encoder signals or drive them to a software defined state. When Motor Channel Encoder is selected in the *Mode* parameter, Channel 1 is routed to OP6, OP5 and OP4, Channel 2 is routed to OP9, OP8 and OP7, and Channel 3 is routed to OP12, OP11 and OP10.

This command is used to configure the IO on the 37 Pin AUX IO connector on the rear panel of the BBD30x units. The connectors can be associated with a specific motor channel and be set to an input or an output as follows.

SET
Command structure (12 bytes)
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
66	02	0C	00	d	s	OutPort		Mode		SwSource	

Data Structure:

field	description	format
OutPort	AUX I/O port number being configured (bitwise OR'ed to set multiple ports simultaneously PORTNUM_1 0x0001 // Aux output port number 1 PORTNUM_2 0x0002 // Aux output port number 2 PORTNUM_3 0x0004 // Aux output port number 3	word
Mode	The operating mode of the associated IO port: SW 0x0001 Aux output(s) are controlled by software ENC 0x0002 Aux outputs are driven by encoder corresponding to the motor channel	word
SwSource	The software state of the port, i.e., High = 0x01, Low = 0x02	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
67	02	Chan Ident	OutPort	d	s

Byte 3 of the request message specifies the port number being addressed, e.g. AUX IO #3 = 0x04.

Example: Request the settings for IO Port BNC 3

TX 67, 02, 01, 04, 50, 01

GET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_ANALOGMONITORCONFIG 0x0269
MGMSG_MOT_REQ_ANALOGMONITORCONFIG 0x0270
MGMSG_MOT_GET_ANALOGMONITORCONFIG 0x0271

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: When the BNC #1 or BNC#2 connector on the rear panel is configured to be an analog output (using the SET_IOCONFIG message), the BNC I/O port can be used to output an analogue voltage in the range of 0 to +5V. This can be assigned to a number of system variables and provides a very low latency way of monitoring the state of the system as the signal follows any changes to the system variable at the speed of the hardware. For example, absolute position can be monitored to aid PID tuning.
This message is used to configure the analog output.

SET

Command structure (20 bytes)

6 byte header followed by 14 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
69	02	0C	00	d	s	Monitor		MotorChannel		SysVar	

12	13	14	15	16	17	18	19
Data							
Scale				Offset			

Data Structure:

field	description	format
Monitor	Analogue monitor number (1 or 2 for the BBD30x)	word
MotorChannel	The Motor channel number associated with the analog output: MOTCHAN1 0x01 The output is monitoring Motor Channel #1 MOTCHAN2 0x02 The output is monitoring Motor r Channel #2	word
SysVar	The system variable to be monitored: POSERROR 0x01 Position error (with SCALE and OFFSET) POSITION 0x02 Actual position (with SCALE and OFFSET) IPHASEA 0x03 Motor phase A current (absolute scale) IPHASEB 0x04 Motor phase B current (absolute scale) IPHASEC 0x05 Motor phase V current (absolute scale) IMOT 0x06 Motor current (absolute scale)	word
Scale		long
Offset		long

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
70	02	Chan Ident	00	d	s

Example: Request the analog output settings for motor channel #1

TX 70, 02, 01, 00, 50, 01

GET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_POSTRIGENSTATE 0x0272
MGMSG_MOT_REQ_POSTRIGENSTATE 0x0273
MGMSG_MOT_GET_POSTRIGENSTATE 0x0274

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: When the rear panel BNC connector on the rear panel is configured to output a position trigger option (set using the SET_MOTTRIGIOCONFIG message), this message Sets, Gets and Requests the position triggering state, i.e. arms or cancels the position trigger engine.
This message must be called to arm the position trigger before it can be used.
Note if the step parameters are changed when the position trigger is enabled a reset to the enabled state should be performed by calling this message to disarm the trigger and then calling it again to arm it.

SET
Command structure
6 byte header only.

0	1	2	3	4	5
Header Only					
72	02	ChanID	State	d	s

Data Structure:

field	description	format
Channel Ident	The associated motor channel number #1, #2 or #3	word
MotorChannel	The position trigger state:: TRIG_ARM 0x01 Arms the position trigger engine TRIG_CANCEL 0x02 Cancels any ongoing position triggering	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
73	02	Chan Ident	00	d	s

Example: Request the analog output settings for motor channel #1

TX 70, 02, 01, 00, 50, 01

GET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_LCDDISPLAYPARAMS 0x0543
MGMSG_MOT_REQ_LCDDISPLAYPARAMS 0x0544
MGMSG_MOT_GET_LCDDISPLAYPARAMS 0x0545

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: This message sets various parameters relating to the front panel display.

SET
Command structure (16 bytes)
6 byte header followed by 10 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
43	05	0C	00	d	s	JSSensitivity		DispBrightness		DispTimeOut	

12	13	14	15
Data			
DispDimLevel		Reserved	

Data Structure:

field	description	format
JSSensitivity	The direction sense and scaling factor (-32768 to +32767) of the knob on the front panel of the unit.	word
DispBrightness	The display brightness when the unit is active (0 to 100 => 0% to 100% brightness)	word
DispTimeOut	Display timeout in minutes (a static display will dim after this interval has elapsed).	word
DispDimLevel	After a certain time (entered in the Time Out parameter above) the display will dim to avoid burn out. This parameter sets the dim level as a percentage of full brightness (range: 0 to 100 but limited to wDispBrightness).	word
Reserved		word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
44	05	Chan Ident	00	d	s

Example: Request the LCD display settings1

TX 44, 05, 01, 00, 50, 01

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_LCDMOVEPARAMS 0x0546
MGMSG_MOT_REQ_LCDMOVEPARAMS 0x0547
MGMSG_MOT_GET_LCDMOVEPARAMS 0x0548

This message is applicable only to BBD301, BBD302 and BBD303 units

Function: This message sets various parameters relating to moves initiated via the front panel.

SET

Command structure (30 bytes)

6 byte header followed by 24 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
46	05	0C	00	d	s	Channel		JSMode		JogStepSize			

14	15	16	17	18	19	20	21	22	23	24	25	26	27
Data													
Accn				MaxVel				JogStopMode		PresetPos			

28	29
Data	
Reserved	

Data Structure:

field	description	format
Channel	The channel being addressed (0x01, 0x02 or 0x03) encoded as a 16-bit word (e.g., 0x01 0x00)	word
JSMode	LCD knob control mode (velocity or single step) VELOCITY 0x01 LCD control knob in velocity control mode. JOG 0x02 LCD control knob for jogging in discrete steps, defined by JogStepSize.	word
JogStepSize	LCD control knob initiated jog step size (in position steps, only used in single step mode)	long
Accn	Acceleration in position pos. steps/sec/sec for all LCD display board initiated moves	long
MaxVel	Maximum velocity in pos. steps/sec for all LCD display board initiated moves	long
JogStopMode	The stop mode defines either an immediate (abrupt) or profiled tops. Set this byte to 0x01 to stop immediately, or to 0x02 to stop in a controller (profiled) manner.	Word
PresetPos	Preset (teach) positions in wheel 'GoTo Position' mode [in position steps	long
Reserved		word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
47	05	Chan Ident	00	d	s

Example: Request the LCD move parameters1

TX 47, 05, 01, 00, 50, 01

GET:
Command structure (30 bytes)
6 byte header followed by 24 byte data packet.
For structure see SET message above.

MGMSG_MOT_SET_MOVESYNCHARRAY 0x0A00

Function: This command defines a time-position array.
The command is different from the majority of other Thorlabs commands in that its length is variable. This is necessary to cater for time-position arrays of different lengths. The number of time-position points contained in this message is indicated by the parameter *wNumPoints* and the actual time-position array sent in the message contains the corresponding number of array rows in the *lTimePos[]* part of the message. Furthermore, in order to constrain the packet size, the maximum number of data points is in a single message is limited to 256.
If the time-position array is longer than this, it must be packaged into a series of messages, with the start index parameter *wStartIx* adjusted accordingly. Similarly, the time position array is limited to a total of 60,000 data points.
As the time-position array may contain position points for any number of channels up to the number supported by the controller, the *wChannels* parameter indicates which channels the position data is for.
To support downloading several different time-position arrays, the parameter *wArrayID* identifies the array.
Note. The current generation of BBD30X controllers only supports a single array and therefore, this parameter must be set to “1” currently. This field is to allow for future development to extend this feature.
The time points are encoded differentially, i.e. the value indicates the time difference between the last and the current time.
All values are in machine units and the conversion factors listed in Section 8 (Conversion between position, velocity and acceleration values in standard physical units and their equivalent Thorlabs parameters) at the beginning of this document.

SET:

Command structure (Variable Length):

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header only						*Data*							
80	09	xx	xx	D0	S1	ArrayID		Channels		NumPoints		StartIx	

14	15	16	17	18	19	20	21	22	23	24	25	Etc…	Etc…
Data													
TimePos[]													

Data Structure:

field	description	format
ArrayID	The array being addressed. This supports the downloading of several different position arrays. Note. The current generation of BBD30X controllers only supports a single array and therefore, this parameter must be set to “1” currently. This field is to allow for	word

	future development to extend this feature.	
Channels	Bitwise OR of all channels in the time position array.	word
NumPoints	The number of Time Position points contained in the message, maximum 256.	word
StartIx	The start index parameter for the array (zero-based). This is used if the time-position array is longer than 256 data points, and therefore must be packaged into a series of messages.	word
TimePos[]	The data for the time position array. For example… In the example below we will consider two cases. If the time-position array involves two motor channels, then TimePos[] will be Time[0], Pos1[0], Pos2[0], Time[1], Pos1[1], Pos2[1], Time[2], Pos1[2], Pos2[2], …. etc – effectively a sequence of three 32-bit values. With 3 channels involved then TimePos[] would be Time[0], Pos1[0], Pos2[0], Pos3[0], Time[1], Pos1[1], Pos2[1], Pos3[1], Time[2], Pos1[2], Pos2[2], Pos3[2] …. Etc The maximum number of data points in the array is 60,000.	Unconstrained array of long

Example:
Assuming the time-position array to be downloaded contains 100 entries for channels 1 and 2 (time + channel 1 position + channel 2 position values), starting from index zero.
The first few entries in the time-position array are shown below:

0	1100000	1350000
80	1100002	1348690
80	1100011	1347867
80	1100023	1347666

The corresponding message is below:
80 09 B8 04 D0 01 01 00 03 00 64 00 00 00 00 00 00 00 E0 C8 10 00 70 99 14 00 50 00 00 00 E2 C8 10 00 52 94 14 00 50 00 00 00 EB C8 10 00 1B 91 14 00 50 00 00 00 F7 C8 10 00 52 90 14 00

80 09 B8 04 D0 01: Message header, indicating 0x04B8 (1208) bytes to follow the header.

Out of these 1208 bytes, the first 8 contains the wArrayID (2 bytes) + wChannels (2 bytes) + wNumPoints (2 bytes) + wStartIx (2 bytes) parameters.
The remaining 1200 bytes contain the 100 entries for each time-position entry.
As the lTimePos[] array contains three 4-byte values, each entry in the array requires 12 bytes, therefore 100 points requires 1200 bytes.

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	Etc…
TimePos[]																
Time[0]				Pos1[0]				Pos2[0]				Time[1]				Etc…

Note that the destination byte of the message (D0) indicates that the message is sent to the whole controller, rather than any individual channel. With multi-axis synchronized moves, sending the command to an individual channel would be in contradiction with the purpose of the command.

01 00: wArrayID array ID number 1
03 00: wChannels 0x01 | 0x02, indicating that the time-position array data is for channels 1 and 2
64 00: wNumPoints this message contains 100 time-position entries (array elements)
00 00: wStartIx the start index for the entries is zero

00 00 00 00 E0 C8 10 00 70 99 14 00: the first time + channel 1 position + channel 2 position values: 0, 1100000, 1350000
50 00 00 00 E2 C8 10 00 52 94 14 00: the second entry: 80, 1100002, 1348690
And so forth.

Note that if the array was longer than 100 time-position points, the next message would have the *wStartIx* parameter set to 100, as the first message contained data for array indices 0 to 99. The user must take care to ensure the continuity of the data, as leaving gaps will result in unpredictable behaviour.

MGMSG_MOT_SET_MOVESYNCHPARAMS 0x0A03

Function: This command specifies the parameters for outputting the time-position array.
As explained in the section AN INTRODUCTION TO MULTI-AXIS SYNCHRONIZED MOVES, the time-position array contains a Leading Section, a Repeated Section and a Trailing Section. The command downloads these parameters. Thus, the time-position array will be output with the repeated section beginning at *wCycleStartIx*, and finishing at *wCycleEndIx*, and repeating *wNumCycles* number of times.
In line with the command defining the time-position array, the parameter *wArrayID* identifies the array that the parameters above are applied to.
Normally, the time-position array lead-in and lead-out sections contain a smooth transition from stationary to moving state and then back to stationary again, as the multi-axis synchronized move is assumed to describe a complete move sequence. However, if the move needs to be interrupted and stopped, the parameter *lDeceleration* will be applied to bring the various moving axes to standstill.

SET:

Command structure (Variable Length):

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header only						*Data*							
83	09	xx	xx	D0	S1	ArrayID		CycleStartIx		CycleEndIx		NumCycles	

14	15	16	17	18	19	20	21	22	23	24	25
Data											
EndIx		Deceleration				Reserved		Reserved		Reserved	

Data Structure:

field	description	format
ArrayID	The array being addressed. This supports the downloading of several different position arrays	word
CycleStartIx	Start index of repeated (cyclic) part of the trajectory (zero-based)	word
CycleEndIx	End index of cyclic part (zero-based)	word
NumCycles	Number of times the cyclic part is repeated	word
EndIx	End index of the complete synchronized move sequence (zero-based) i.e. total number of time-position array elements. The BBD30X (currently) has a maximum capacity of 60,000.	word
Deceleration	Deceleration time if the move is stopped before completing all points. This measured in controller-specific servo update intervals	long
Reserved		word
Reserved		word

Reserved		word

Example:
Assuming the synchronous move parameters as follows:
Cycle to start at index 4, end at index 1207, repeated 3 times, with the complete section finishing at index 1211, using deceleration value of 6871 for stop.

83 09 26 00 D0 01 01 00 04 00 B7 04 03 00 BB 04 D7 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The header of the message (83 09 26 00 D0 01) follows the same format as before. The bytes that follow contain the parameters:

01 00: wArrayID array ID number 1
04 00: wCycleStartIx the repeated section of the array starts at index 4
B7 04: wCycleEndIx the repeated section of the array ends at index 1207
03 00: wNumCycles the repeated section is repeated 3 times
BB 04: wEndIx the end of the entire sequence is at index 1211
D7 1A 00 00: lDeceleration use a deceleration value of 6871 if the move is stopped

The remaining bytes are reserved and sent as zeros.

MGMSG_MOT_MOVE_SYNCHSTART 0x0A06

Function: This command is used to start a synchronized multi-axis move that has been defined with the previous two commands (0x0980 and 0x0983).
In line with the command defining the time-position array, the parameter *wArrayID* identifies the array that the parameters above are applied to.
The *wChannels* parameter defines which channels to start. Normally this would contain all the channels that the time-position array was downloaded for, although it is also possible to start the move only on some of the channels.
The *wTrigger* parameter defines the trigger for starting the synchronized move. If this parameter is 0x01, the move will start immediately (software trigger).

SET:
Command structure (Variable Length):

0	1	2	3	4	5	6	7	8	9	10	11
header only						*Data*					
83	09	xx	xx	D0	S1	ArrayID		Channels		Trigger	

Data Structure:

field	description	format
ArrayID	The array being addressed. This supports the downloading of several different position arrays	word
Channels	Bitwise OR of all channels in the time position array.	word
Trigger	Trigger source to start the move as follows: 1 – Software Trigger	word

Example:
Start the multi-axis synchronized move on channels 1 and 2
86 09 06 00 D0 01 01 00 03 00 01 00
The header of the message (86 09 06 00 D0 01) follows the same format as before. The bytes that follow contain the parameters:

01 00: wArrayID array ID number 1.
03 00: wChannels start the multi-axis synchronized move on channels 1 and 2.
01 00: wTrigger software trigger, the move is started immediately.

Note: to stop a synchronized multi-axis move the MGMSG_MOT_MOVE_STOP (0x0465) command can be used, with the channel idents bitwise OR'ed.

MGMSG_MOT_SET_RASTERMOVEPARAMS 0x0A10
MGMSG_MOT_REQ_RASTERMOVEPARAMS 0x0A11
MGMSG_MOT_GET_RASTERMOVEPARAMS 0x0A12

Function: This command specifies parameters that define the raster pattern performed when the MOVERASTER message is called.
.

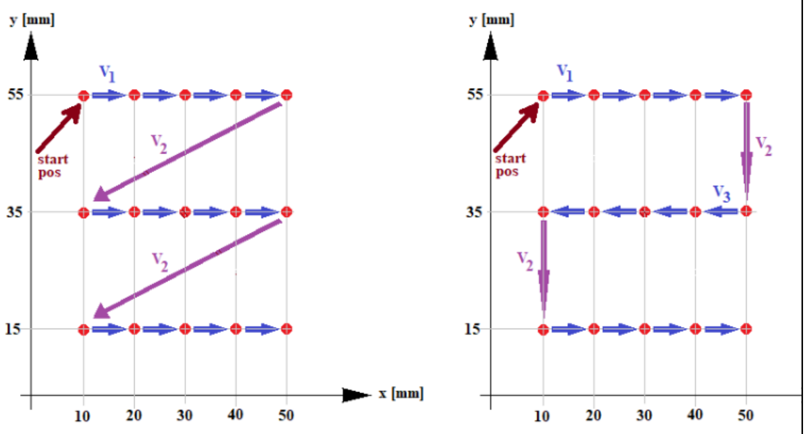
SET:
Command structure (Variable Length):

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header only						*Data*							
10	0A	xx	xx	D0	S1	ScanPattern		TrtiggerSource		TriggerMode		TriggerPolarity	

14	15	16	17	18	19	20	21	22	23	24	25	26	27
Data													
StartPosX				RelDistanceX				NumCyclesX				DwellTimeX	

28	29	30	31	32	33	34	35	36	37	38	39	40	41
Data													
StartPosY				RelDistanceY				NumCyclesY				DwellTimeX	

Data Structure:

field	description	format
ScanPattern	Defines the type of raster scan pattern to be performed: flyback/unidirectional scan (0x01) or reverse/bidirectional scan0x02) 0x01 - FLYBACK Flyback (unidirectional) scan 0x02 - FWDREV Forward/Reverse (bidirectional) scan  Flyback scan Forward-reverse scan	word
TriggerSource	Sets the trigger source as follows: 0x00 - The motor channel's trigger input source is software 0x01 - The motor channel's trigger input source is BNC #1 0x02 - The motor channel's trigger input source is BNC #2 0x03 - The motor channel's trigger input source is BNC #3	word
TriggerMod	Sets the trigger mode as follows:	word

e	0x01 - SOFTWARE Software trigger, raster scan move starts immediately 0x02 - XSTEP Trigger starts next relative move along the X axis 0x03 - YSTEP Trigger starts the complete X axis move sequence, then stops and awaits next trigger before Y step 0x04 - XYSCAN Trigger starts the complete sequence of all moves 0x05 - ONOFF Trigger starts the whole sequence, stops when trigger is de-asserted, resumes when trigger is asserted	
TriggerPolarity	Sets the trigger polarity as follows: 0x01 - Trigger polarity High 0x02 - Trigger polarity Low	word
StartPosX	The start position for the X axis in mm	long
RelDistanceX	Positive or negative relative distance to move (in position steps).	long
NumCyclesX	Number of times the relative move is performed	long
DwellTimeX	Dwell time in milliseconds	word
StartPosY	The start position for the Y axis in mm	long
RelDistanceY	Positive or negative relative distance to move (in position steps).	long
NumCyclesY	Number of times the relative move is performed	long
DwellTimeY	Dwell time in milliseconds	word

Example:
Configure a raster scan with the parameters detailed below:
Scan Pattern: *Flyback*
Trigger Source: *BNC2*
Trigger Mode: *X Step*
Trigger Polarity: *High*
Start Position X: *15 mm*
Relative Distance X: *10 mm*
Number of Cycles X: *5*
Dwell Time X: *500 ms*
Start Position Y: *20 mm*
Relative Distance Y: *10 mm*
Number of Cycles Y: *6*
Dwell Time Y: *1000 ms*

In the code example below, the message header is in red, parameters for the whole scan are in purple, X-axis parameters are in blue and Y-axis parameters are in green
10 0A 24 00 91 01 01 00 02 00 02 00 01 00 E0 93 04 00 40 0D 03 00 05 00 00 00 F4 01 80 1A 06 00 40 0D 03 00 06 00 00 00 E8 03

The header of the message (10 0A 24 00 91 01) follows the normal format. The bytes that follow contain the parameters:

01 00 Scan Pattern: *Flyback*
02 00 Trigger Source: *BNC2*
02 00 Trigger Mode: *X Step*
01 00 Trigger Polarity: *High*
E0 93 04 00 Start Position X: *15 mm*
40 0D 03 00 Relative Distance X: *10 mm*
05 00 00 00 Number of Cycles X: *5*

F4 01 Dwell Time X: *500 ms*
80 1A 06 00 Start Position Y: *20 mm*
40 0D 03 00 Relative Distance Y: *10 mm*
06 00 00 00 Number of Cycles Y: *6*
E8 03 Dwell Time Y: *1000 ms*

Note: to start, pause or stop a raster scan the MGMSG_MOT_MOVE_RASTER (0x0A13) command can be used.

MGMSG_MOT_MOVE_RASTER 0x0A13

Function: This command is used to start, stop and pause a raster scan that has been defined with the previous command (0x0A10).
Note that if the raster move is hardware triggered, the command will only act as an enable/disable signal and the actual move will also be subject to the hardware trigger state.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	0A	Chan Ident	Command	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	char
COMMAND	The command as a 4 bit integer: 0x01 START – Starts the raster scan. 0x02 PAUSE – Pauses the raster scan. The move will continue after the next start command. 0x03 STOP – Stops the raster scan. The next start command will re-start the raster scan from the beginning.	char

Example: Start the raster scan'.

TX 13, 0A, 01, 01, 50, 01

13, 0A MOVE_RASTER
01, Channel 1
01, Start the raster scan
50, destination Generic USB device
01, Source PC

Filter Flipper Control Messages

Introduction

The Thorlabs Filter Flipper drive uses the Motor server control instance control its functionality. The messages listed here provide the extra functionality required for a client application to control one or more of the Thorlabs series of MFF series flipper units.

MGMSG_MOT_SET_MFF_OPERPARAMS 0x0510
MGMSG_MOT_REQ_MFF_OPERPARAMS 0x0511
MGMSG_MOT_GET_MFF_OPERPARAMS 0x0512

Function: Used to set various operating parameters that dictate the function of the MFF series flipper unit.

SET:

Command structure (40 bytes)
6 byte header followed by 34 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
10	05	22	00	d|	s	Chan Ident		lTransitTime			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
lTransitTimeADC				OperMode1		SigMode1		PulseWidth1			

24	25	26	27	28	29	30	31	32	33	34	35
				Data							
OperMode2		SigMode2		PulseWidth2				Not Used			

36	37	38	39
Not Used			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
lTransitTime	The time taken (in milliseconds) for the flipper to move from position 1 to position 2 and vice versa. Values must be entered in the range 300 to 2800 ms.	long
lTransitTimeADC	The time taken (in ADC counts) for the flipper to move from position 1 to position 2 and vice versa. The number of ADC counts is calculated from an equation that relates actual time of flight in milli-seconds to the ADC value required by the flipper code. The equation relating the two variables is defined as follows TransitTimeADC = 10000000 x $\text{TransitTime}^{-1.591}$ Example A transit time of 500 ms would be calculated as TransitTimeADC = 10000000 x $500^{-1.591}$ = 10000000 x 0.00005080877 = 508.0877 so a user requiring 500ms motion time needs to set 508 as the ADC value in the structure. This value is then used by the flipper to give a reasonable approximation for the actual time of flight.	long
wDigIO1OperMode	Specifies the operating mode of the DIG IO 1	word

	input/output signal as follows: 01 Sets IO connector to input and 'toggle position' mode. In this mode, the input signal causes flipper to move to other position). 02 Sets IO connector to input and 'goto position' mode. In this mode, the input signal dictates flipper position, POS 1 or POS 2. as dictated by the Button Input or Button Input (Swap Pos) parameters set in the DigIOSigMode parameter below. 03 Sets IO connector to output mode, where the O/P signal indicates the flipper is 'at position'. 04 Sets IO connector to output mode, where the O/P signal indicates the flipper is in motion (i.e. between positions).	
wDigIO1SigMode	Specifies the functionality of the input/output signal. as follows: 01 The connector can be short circuited (e.g. with button). If the Operating Mode is set to Input:Toggle Position then a short circuit causes the flipper to toggle position. If the Operating Mode is set to Input: Goto Position then a short circuit causes the flipper to move to Pos 1 and open circuit causes flipper to move to POS 02. The connector is set to logic input where a logic transition (edge) dictates flipper operation. If the Operating Mode above set to Input:Toggle Position, then a LO to HI edge causes flipper to toggle position. If the Operating Mode is set to Input: Goto Position, then a LO to HI edge causes the flipper to move to POS 1 and a HI to LO edge causes the flipper to move to POS 2. 04 This parameter can be 'Bitwise Ored' with either the button or the logic parameters above, such that the open circuit and short circuit or the edge functionality is swapped. 10 The connector is set to a logic output where the logic transition (edge) represents flipper position. If the Operating Mode above is set to Output: At Position, then a LO to HI edge (HI level) indicates flipper is at POS 1 and a HI to LO edge (LO level) indicates the flipper is at POS 2. If the Operating Mode above is set to Output: InMotion, then a LO to HI edge (HI level) indicates the flipper is moving between positions and a HI to LO edge (LO level) indicates the flipper has stopped moving. 20 MFFSIGMODE_OP_PULSE The connector is set to	word

	a logic output where a logic pulse indicates flipper operation. If the Operating Mode above is set to Output: At Position, then a logic HI pulse indicates flipper has reached a position. If the Operating Mode above is set to Output: InMotion, then a logic HI pulse indicates the flipper has started moving. The Pulse width is set in the Signal Width paramter below. 40 This parameter can be 'Bitwise Ored' with either the level (edge) or the pulse parameters above, such that the level or pulse functionality is swapped.	
lDigIO1PulseWidth	The pulse width in ms when the Digital Signal Mode described previously is set to Logic Pulse Output or Logic Pulse Output (Inverted). The pulse width is set within the range 10 to 200 ms.	long
wDigIO2OperMode	As DigIO1	word
wDigIO2SigMode	As DigIO1	word
lDigIO2PulseWidth	As DigIO1	long
Not Used		long
Not Used		dword

Example: Set the MFF parameters for chan 1 as follows:

TransitTime	500 ms
TransitTimeADC	508 counts
DigIO1OperMode	Toggle Position
DigIO1SigMode	Button Mode Input
DigIO1PulseWidth	200 ms
DigIO2OperMode	Toggle Position
DigIO2SigMode	Button Mode Input
DigIO2PulseWidth	200 ms
Not Used	
Not Used	

TX 10,05,22,00,D0,01,
01,00,F4,01,00,00,FC,01,00,00,01,00,01,00,C8,00,00,00,01,00,01,00,C8,00,00,00,00,00,00,00
,00,00,00,00

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	05	Chan Ident	00	d	s

Example: Request the MFF operating modes

TX 11, 05, 01, 00, 50, 01

GET:

Response structure (40 bytes):

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
10	05	22	00	d|	s	Chan Ident		lTransitTime			

12	13	14	15	16	17	18	19	20	21	22	23
Data											
lTransitTimeADC				OperMode1		SigMode1		PulseWidth1			

24	25	26	27	28	29	30	31	32	33	34	35
				Data							
OperMode2		SigMode2		PulseWidth2				Not Used			

36	37	38	39
Not Used			

See SET for structure

Solenoid Control Messages

Introduction

The Thorlabs Solenoid drive uses the Motor server control instance control its functionality. The messages listed here provide the extra functionality required for a client application to control one or more of the Thorlabs series of TSC001 T-Cube solenoid driver units.

MGMSG_MOT_SET_SOL_OPERATINGMODE 0x04C0
MGMSG_MOT_REQ_SOL_OPERATINGMODE 0x04C1
MGMSG_MOT_GET_SOL_OPERATINGMODE 0x04C2

Function: This message sets the operating mode of the solenoid driver.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C0	04	Chan Ident	Mode	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	char
Operating Mode	The operating mode of the unit as a 4 bit integer: 0x01 SOLENOID_MANUAL - In this mode, operation of the solenoid is via the front panel 'Enable' button, or by the 'Output' buttons on the GUI panel. 0x02 SOLENOID_SINGLE - In this mode, the solenoid will open and close each time the front panel 'Enable' button is pressed, or the 'Output ON' button on the GUI panel is clicked. The ON and OFF times are specified by calling the MGMSG_MOT_SET_SOL_CYCLEPARAMS message. 0x03 SOLENOID_AUTO - In this mode, the solenoid will open and close continuously after the front panel 'Enable' button is pressed, or the 'Output ON' button on the GUI panel is clicked. The ON and OFF times, and the number of cycles performed, are specified by calling the MGMSG_MOT_SET_SOL_CYCLEPARAMS message. 0x04 SOLENOID_TRIGGER - In Triggered mode, a rising edge on rear panel TRIG IN BNC input will start execution of the parameters programmed on the unit (On Time, Off Time, Num Cycles - see MGMSG_MOT_SET_SOL_CYCLEPARAMS message.). The unit must be primed (i.e. the ENABLE button pressed and the ENABLED LED lit) before the unit can respond to the external trigger.	char

Example: Set the control mode to 'Single'.

TX C0, 04, 01, 02, 50, 01

C0,04 SET_SOL_OPERATINGMODE
01, Channel 1
02, Set mode to 'Single'
50, destination Generic USB device
01, Source PC

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	04	Chan Ident	00	d	s

Example: Request the control mode

TX C1, 04, 01, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
C2	04	Chan Ident	Mode	d	s

Example: Get the control mode currently set.

RX C2, 04, 01, 01, 01, 50

MGMSG_MOT_SET_SOL_CYCLEPARAMS 0x04C3
MGMSG_MOT_REQ_SOL_CYCLEPARAMS 0x04C4
MGMSG_MOT_GET_SOL_CYCLEPARAMS 0x04C5

Function: Used to set the cycle parameters that are applicable when the solenoid controller is operating in one of the non-manual modes.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C3	04	0E	00	d|	s	Chan Ident		OnTime			

12	13	14	15	16	17	18	19
Data							
OffTime				NumCycles			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
OnTime	The time which the solenoid is activated (100ms to 10,000s in 1 ms steps)	long
OffTime	The time which the solenoid is a de-activated (100ms to 10,000s in 1 ms steps)	long
NumCycles	If the unit is operating in 'Auto' mode, the number of Open/Close cycles to perform. (0 to 1,000,000) is specified in the NumCycles parameter. If set to '0' the unit cycles indefinitely. If the unit is not operating in 'Auto' mode, the NumCycles parameter is ignored.	long

Example: Set the cycle parameters parameters for chan 1 as follows:
OnTime: 1000ms
OffTime: 1000ms
NumCycles: 20

TX C3, 04, 0E, 00, D0, 01, 01, 00, E8, 03, 00, 00, E8, 03, 00, 00, 14, 00, 00, 00

Header: C3, 04, 0E, 00, D0, 01: Set Cycle Params, D0H (14) byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1 (always set to 1 for TSC001)
OnTime: *E8, 03, 00, 00:* Set on time to 1000 ms (i.e. 1000 ms)
OffTime: *E8, 03, 00, 00:* Set off time to 1000 ms (i.e. 1000 ms)
NumCycles: *14, 00, 00, 00:* Set number of cycles to 20

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
C4	04	Chan Ident	00	d	s

GET:

Response structure (20 bytes)

6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C5	04	0E	00	d|	s	Chan Ident		OnTime			

12	13	14	15	16	17	18	19
Data							
OffTime				NumCycles			

For structure see SET message above.

MGMSG_MOT_SET_SOL_INTERLOCKMODE 0x04C6
MGMSG_MOT_REQ_SOL_INTERLOCKMODE 0x04C7
MGMSG_MOT_GET_SOL_INTERLOCKMODE 0x04C8

Function: The solenoid unit features a hardware interlock jackplug. This message specifies whether the solenoid driver requires the hardware interlock to be fitted before it can operate.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
C6	04	Chan Ident	Mode	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	char
Interlock Mode	The operating mode of the unit as a 4 bit integer: 0x01 SOLENOID_ENABLED – The hardware interlock must be fitted before the unit can be operated. 0x02 SOLENOID_DISABLED – The hardware interlock is not required.	char

Example: Set the interlock mode to 'Enabled'.

TX C6, 04, 01, 01, 50, 01

C0,06 SET_SOL_INTERLOCKMODE
01, Channel 1
01, Set mode to 'Enabled'
50, destination Generic USB device
01, Source PC

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C7	04	Chan Ident	00	d	s

Example: Request the control mode

TX C7, 04, 01, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
C8	04	Chan Ident	Mode	d	s

Example: Get the control mode currently set.

RX C8, 04, 01, 01, 01, 50

MGMSG_MOT_SET_SOL_STATE 0x04CB
MGMSG_MOT_REQ_SOL_STATE 0x04CC
MGMSG_MOT_GET_SOL_STATE 0x04CD

Function: This message sets the output state of the solenoid unit, and overrides any existing settings. It can also be operated by the SET_CHANENABLESTATE message.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
CB	04	Chan Ident	State	d	s

Data Structure:

field	description	format
Chan Ident	The channel being addressed	char
Interlock Mode	The operating mode of the unit as a 4 bit integer: 0x01 SOLENOID_ON – The solenoid is active. 0x02 SOLENOID_OFF – The solenoid is de-activated.	char

Example: Set the solenoid to ‘ON’.

TX CB, 04, 01, 01, 50, 01

CB,06 SET_SOL_STATE
01, Channel 1
01, Set state to ‘ON’
50, destination Generic USB device
01, Source PC

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
CC	04	Chan Ident	00	d	s

Example: Request the control mode

TX CC, 04, 01, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
CD	04	Chan Ident	Mode	d	s

Example: Get the control mode currently set.

RX CD, 04, 01, 01, 01, 50

Piezo Control Messages

Introduction

The ‘Piezo’ control messages provide the functionality required for a client application to control one or more of the Thorlabs series of piezo controller units. This range of controllers covers both open and closed loop piezo control in a variety of formats including compact Cube type controllers, benchtop units and 19” rack based modular drivers. **Note.** For ease of description, the TSG001 T-Cube Strain Gauge reader is considered here as a piezo controller.

The piezo messages can be used to perform activities such as selecting output voltages, reading the strain gauge position feedback, operating open and closed loop modes and enabling force sensing mode. With a few exceptions, these messages are generic and apply equally to both single and dual channel units.

Where applicable, the target channel is identified in the lChanID parameter and on single channel units, this must be set to CHAN1_ID. On dual channel units, this can be set to CHAN1_ID, CHAN2_ID or CHANBOTH_ID as required.

For details on the operation of the Piezo Controller, and information on the principles of operation, refer to the handbook supplied with the unit.

MGMSG_PZ_SET_POSCONTROLMODE 0x0640
MGMSG_PZ_REQ_POSCONTROLMODE 0x0641
MGMSG_PZ_GET_POSCONTROLMODE 0x0642

Function: When in closed-loop mode, position is maintained by a feedback signal from the piezo actuator. This is only possible when using actuators equipped with position sensing.
This method sets the control loop status The Control Mode is specified in the Mode parameter as follows:

0x01 Open Loop (no feedback)
0x02 Closed Loop (feedback employed)
0x03 Open Loop Smooth
0x04 Closed Loop Smooth

If set to Open Loop Smooth or Closed Loop Smooth is selected, the feedback status is the same as above however the transition from open to closed loop (or vise versa) is achieved over a longer period in order to minimize voltage transients (spikes).

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
40	06	Chan Ident	Mode	d	s

Example: Set the control mode to closed loop.

TX 40, 06, 01, 02, 50, 01

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	06	Chan Ident	00	d	s

Example: Request the control mode

TX 41, 06, 01, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5
header only					
42	06	Chan Ident	Mode	d	s

Example: Get the control mode that is currently set.

RX 42, 06, 01, 02, 01, 50

MGMSG_PZMOT_MOVE_START 0x2100

Function: Used to start an open/close loop move.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
00	21	ChanIdent	StartMove	d	s

Data Structure:

field	description	format
ChanIdent	The channel being addressed	char
StartMove	Move start control, 0x1 start move, others don't move.	char

Upon completion of the move the controller sends a Move Completed message as previously described.

Example: Move start with channel 1.
TX 00, 21, 01, 01, 50, 01

MGMSG_PZMOT_PULSE_PARA_ACQUIRE 0x2102
MGMSG_PZMOT_ PULSE_PARA_ACQUIRED 0x2103

Function: Used to start pulse parameters acquire operation for optimized performance.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
02	21	ChanIdent	00	d	s

Data Structure:

field	description	format
ChanIdent	The channel being addressed	char

Upon completion of the move, the controller sends an acquired parameters operation finished message (MGMSG_PZMOT_PULSE_PARA_ACQUIRED).

Example: Start parameters acquire operation with channel 1 to get the best performance parameters.
TX 02, 21 ,01 ,00 ,50 ,01

ACQUIRED:

Function: No response on initial message, but upon completion of parameters acquire operation sends an “acquired parameters” message:

RX structure (6 bytes):

0	1	2	3	4	5
header only					
03	21	ChanIdent	00	d	s

Example: The best performance parameters for channel 1 has been acquired.
TX 03, 21 ,01 ,00 ,01 ,50

MGMSG_PZ_SET_OUTPUTVOLTS 0x0643
MGMSG_PZ_REQ_OUTPUTVOLTS 0x0644
MGMSG_PZ_GET_OUTPUTVOLTS 0x0645

Function: Used to set the output voltage applied to the piezo actuator. This command is applicable only in Open Loop mode. If called when in Closed Loop mode it is ignored.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
43	06	04	00	d|	s	Chan Ident		Voltage	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Voltage	The output voltage applied to the piezo when operating in open loop mode. The voltage is set in the range -32768 to 32767 (-7FFF to 7FFF) to which corresponds to -100% to 100% of the maximum output voltage as set using the TPZ_IOSETTINGS command.	short

Example: Set the drive voltage to 70V

TX 43, 06, 04, 00, D0, 01, 01, 00, 77, 77,

Header: 43, 06, 04, 00, D0, 01: SetPZOutputVolts, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
Voltage: *77, 77:* corresponds to 70 V (30583) for a max 75 V unit

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
44	6	Chan Ident	00	d	s

GET:
Response structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
45	06	04	00	d|	s	Chan Ident		Voltage	

For structure see SET message above.

MGMSG_PZ_SET_OUTPUTPOS 0x0646
MGMSG_PZ_REQ_OUTPUTPOS 0x0647
MGMSG_PZ_GET_OUTPUTPOS 0x0648

Function: Used to set the output position of piezo actuator. This command is applicable only in Closed Loop mode. If called when in Open Loop mode it is ignored. The position of the actuator is relative to the datum set for the arrangement using the ZeroPosition method.

SET:

Command structure (10 bytes)

6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
46	06	04	00	d|	s	Chan Ident		PositionSW	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
PositionSW	The output position of the piezo relative to the zero position. The voltage is set as a signed 16-bit integer (word) in the range 0 to 32767 (0 to 7FFF). This corresponds to 0 to 100% of the maximum piezo extension. The negative range (0x800 to FFFF) is not used to set a position at this time. When reporting the position near to 0% of extension, the value may be a small negative value (short).	word

Example: Set the drive position to 15 µm (when total travel = 100 µm).

TX 46, 06, 04, 00, D0, 01, 01, 00, 33, 13,

Header: 46, 06, 04, 00, D0, 01: SetPZOutputPos, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
PositionSW: *33, 13:* corresponds to 15 µm for a max 100 µm unit

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
47	06	Chan Ident	00	d	s

GET:

Response structure (10 bytes)

6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
48	06	04	00	d|	s	Chan Ident		PositionSW	

MGMSG_PZ_SET_INPUTVOLTSSRC 0x0652
MGMSG_PZ_REQ_INPUTVOLTSSRC 0x0653
MGMSG_PZ_GET_INPUTVOLTSSRC 0x0654

Function: Used to set the input source(s) which controls the output from the HV amplifier circuit (i.e. the drive to the piezo actuators).

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
52	06	04	00	d|	s	Chan Ident		VoltSrc	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
VoltSrc	The following values are entered into the VoltSrc parameter to select the various analog sources. *0x00 Software Only*: Unit responds only to software inputs and the HV amp output is that set using the SetVoltOutput method or via the GUI panel. *0x01 External Signal*: Unit sums the differential signal on the rear panel EXT IN (+) and EXT IN (-)connectors with the voltage set using the SetVoltOutput method *0x02 Potentiometer*: The HV amp output is controlled by a potentiometer input (either on the control panel, or connected to the rear panel User I/O D-type connector) summed with the voltage set using the SetVoltOutput method. The values can be 'bitwise ord' to sum the software source with either or both of the other source options.	word

Example: Set the input source to software and potentiometer.

TX 52, 06, 04, 00, D0, 01, 01, 00, 02, 00,

Header: 52, 06, 04, 00, D0, 01: SetVoltsSrc, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
VoltSrc: *02, 00:* selects software and potentiometer inputs

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
53	06	Chan Ident	00	d	s

GET:
Response structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
54	06	04	00	d|	s	Chan Ident		VoltsSrc	

For structure see SET message above.

MGMSG_PZ_SET_PICONSTS 0x0655
MGMSG_PZ_REQ_PICONSTS 0x0656
MGMSG_PZ_GET_PICONSTS 0x0657

Function: Used to set the proportional and integration feedback loop constants. These parameters determine the response characteristics when operating in closed loop mode.
The processors within the controller compare the required (demanded) position with the actual position to create an error, which is then passed through a digital PI-type filter. The filtered value is used to develop an output voltage to drive the piezo.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
55	06	06	00	d|	s	Chan Ident		PropConst		IntConst	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
PropConst	The value of the proportional term in the range 0 to 255.	word
IntConst	The value of the Integral term.in the range 0 to 255	word

Example: Set the PI constants for a TPZ001 unit.

TX 55, 06, 06, 00, D0, 01, 01, 00, 64, 00, 0F, 00

Header: 55, 06, 05, 00, D0, 01: SetPIConsts, 06 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
PropConst: *64, 00:* sets the proportional constant to 100
IntConst: *0F, 00:* sets the integral constant to15

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
56	06	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
57	06	06	00	d|	s	Chan Ident		PropConst		IntConst	

For structure see SET message above.

MGMSG_PZ_REQ_PZSTATUSBITS 0x065B
MGMSG_PZ_GET_PZSTATUSBITS 0x065C

Function: Returns a number of status flags pertaining to the operation of the piezo controller channel specified in the Chan Ident parameter. These flags are returned in a single 32 bit integer parameter and can provide additional useful status information for client application development. The individual bits (flags) of the 32 bit integer value are described in the following tables.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
5B	06	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
5C	06	06	00	d|	s	Chan Ident		StatusBits			

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
StatusBits	The status bits for the associated controller channel. The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TPZ001 controller

Hex Value	Bit Number	Description
0x00000001	1	Piezo actuator connected (1 - connected, 0 - not connected).
	2 to 4	For Future Use
0x00000010	5	Piezo channel has been zero'd (1 - zero'd, 0 not zero'd).
0x00000020	6	Piezo channel is zeroing (1 - zeroing, 0 - not zeroing).
0x00000040	7 to 8	For Future Use
0x00000100	9	Strain gauge feedback connected (1 - connected, 0 - not connected).
	10	For Future Use
0x00000400	11	Position control mode (1 - closed loop, 0 - open loop).
	12 to 20	For Future Use

BPC series controllers

Hex Value	Bit Number	Description
0x00000001	1	Piezo actuator connected (1 - connected, 0 - not connected).
	2 to 4	For Future Use
0x00000010	5	Piezo channel has been zero'd (1 - zero'd, 0 not zero'd).
0x00000020	6	Piezo channel is zeroing (1 - zeroing, 0 - not zeroing).
0x00000040	7 to 8	For Future Use
0x00000100	9	Strain gauge feedback connected (1 - connected, 0 - not connected).
	10	For Future Use
0x00000400	11	Position control mode (1 - closed loop, 0 - open loop).
	12	For Future Use
Note. Bits 13, 14 and 15 are applicable only to BPC30x series controllers.		
0x00001000	13	Hardware set to 75 V max output voltage
0x00002000	14	Hardware set to 100 V max output voltage
0x00004000	15	Hardware set to 150 V max output voltage
	16 to 20	For Future Use
Note. Bits 21 to 28 (Digital Input States) are only applicable if the associated digital input is fitted to your controller – see the relevant handbook for more details		
0x00100000	21	Digital input 1 state (1 - logic high, 0 - logic low).
0x00200000	22	Digital input 2 state (1 - logic high, 0 - logic low).
0x00400000	23	Digital input 3 state (1 - logic high, 0 - logic low).
0x00800000	24	Digital input 4 state (1 - logic high, 0 - logic low).
0x01000000	25	Digital input 5 state (1 - logic high, 0 - logic low).
0x02000000	26	Digital input 6 state (1 - logic high, 0 - logic low).
0x04000000	27	Digital input 7 state (1 - logic high, 0 - logic low).
0x08000000	28	Digital input 8 state (1 - logic high, 0 - logic low).
	29	For Future Use
0x20000000	30	Active (1 – indicates unit is active, 0 – not active)
0x40000000	31	For Future Use
0x80000000	32	Channel enabled (1 – enabled, 0- disabled)

MGMSG_PZ_REQ_PZSTATUSUPDATE 0x0660
MGMSG_PZ_GET_PZSTATUSUPDATE 0x0661

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the position and status of the controller (for example position and O/P voltage). The messages will be sent by the controller each time the function is called.

NOTE. This message is also returned by the NanoTrak control when it is operating in piezo mode.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
60	06	Chan Ident	00	d	s

GET:
Status update messages are received with the following format:-

Response structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
61	06	0A	00	d|	s	Chan Ident		OPVoltage		Position	

12	13	14	15
Status Bits			

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
OPVoltage	The output voltage applied to the piezo. The voltage is returned in the range -32768 to 32767 (-7FFF to 7FFF) which corresponds to -100% to 100% of the maximum output voltage as set using the TPZ_IOSETTINGS command.	short
Position	The position of the piezo. The position is returned in the range 0 to 32767 (0 to 7FFF) which corresponds to 0 to 100% of the maximum position.	short
Status Bits	The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TPZ001 KPZ101 controller

Hex Value	Bit Number	Description
0x00000001	1	Piezo actuator connected (1 - connected, 0 - not connected).
	2 to 4	For Future Use
0x00000010	5	Piezo channel has been zero'd (1 - zero'd, 0 not zero'd).
0x00000020	6	Piezo channel is zeroing (1 - zeroing, 0 - not zeroing).
0x00000040	7 to 8	For Future Use
0x00000100	9	Strain gauge feedback connected (1 - connected, 0 - not connected).
	10	For Future Use
0x00000400	11	Position control mode (1 - closed loop, 0 - open loop).
	12 to 20	For Future Use

BPC series controllers

Hex Value	Bit Number	Description
0x00000001	1	Piezo actuator connected (1 - connected, 0 - not connected).
	2 to 4	For Future Use
0x00000010	5	Piezo channel has been zero'd (1 - zero'd, 0 not zero'd).
0x00000020	6	Piezo channel is zeroing (1 - zeroing, 0 - not zeroing).
0x00000040	7 to 8	For Future Use
0x00000100	9	Strain gauge feedback connected (1 - connected, 0 - not connected).
	10	For Future Use
0x00000400	11	Position control mode (1 - closed loop, 0 - open loop).
	12 to 20	For Future Use
Note. Bits 21 to 28 (Digital Input States) are only applicable if the associated digital input is fitted to your controller – see the relevant handbook for more details		
0x00100000	21	Digital input 1 state (1 - logic high, 0 - logic low).
0x00200000	22	Digital input 2 state (1 - logic high, 0 - logic low).
0x00400000	23	Digital input 3 state (1 - logic high, 0 - logic low).
0x00800000	24	Digital input 4 state (1 - logic high, 0 - logic low).
0x01000000	25	Digital input 5 state (1 - logic high, 0 - logic low).
0x02000000	26	Digital input 6 state (1 - logic high, 0 - logic low).
0x04000000	27	Digital input 7 state (1 - logic high, 0 - logic low).
0x08000000	28	Digital input 8 state (1 - logic high, 0 - logic low).
	29	For Future Use
0x20000000	30	Active (1 – indicates unit is active, 0 – not active)
0x40000000	31	For Future Use
0x80000000	32	Channel enabled (1 – enabled, 0- disabled)

MGMSG_PZ_ACK_PZSTATUSUPDATE 0x0662

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called “server alive” must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
62	06	00	00	d	s

TX 62, 06, 00, 00, 50, 01

MGMSG_PZ_SET_PPC_PIDCONSTS 0x0690
MGMSG_PZ_REQ_PPC_PIDCONSTS 0x0691
MGMSG_PZ_GET_PPC_PIDCONSTS 0x0692

THIS MESSAGE IS APPLICABLE ONLY TO PPC001, PPC102 and CT1P UNITS

Function: When operating in Closed Loop mode, the proportional, integral and derivative (PID) constants can be used to fine tune the behaviour of the feedback loop to changes in the output voltage or position. While closed loop operation allows more precise control of the position, feedback loops need to be adjusted to suit the different types of focus mount assemblies that can be connected to the system. Due to the wide range of objectives that can be used with the PFM450 and their different masses, some loop tuning may be necessary to optimize the response of the system and to avoid instability.
This message sets values for these PID parameters. The default values have been optimized to work with the actuator shipped with the controller and any changes should be made with caution.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
90	06	0C	00	d|	s	Chan Ident		PIDConstsP		PIDConstsI	

12	13	14	15	16	17
Data					
PIDConstsD		PIDConstsDFC		PIDDerivFilterON	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
PIDConstsP	The value of the Proportional term in the range 0 to 10000 (H2719), default 900	Float
PIDConstsI	The value of the Integral term in the range 0 to 10000 (H2719), default 800	Float
PIDConstsD	The value of the Derivative term in the range 0 to 10000 (H2719), default 90	Float
PIDConstsDFC	The value of the Derivative Low Pass Filter Cut Off Frequency in the range 0 to 10000 (H2719), default 1000	Float
PIDDerivFilterON	Derivative Filter ON (0x01) or OFF (0x02)	Word

Example: Set the PID constants

TX 90, 06, 0C, 00, D0, 01, 01, 00, 84, 03, 20, 03, 5A, 00, E8, 03, 01, 00

Header: 90, 06, 0C, 00, D0, 01: SetPIConsts, 12 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
PIDConstsP: *84, 03:* sets the proportional constant to 900
PIDConstsI: *20, 03:* sets the integral constant to 800
PIDConstsD: *5A, 00:* sets the derivative constant to 90
PIDConstsD: *E8, 03:* sets the derivative cut off frequency to 1000
PIDConstsD: *01, 00:* sets the derivative cut off filter ON.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
91	06	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
92	06	0C	00	d|	s	Chan Ident		PIDConstsP		PIDConstsI	

12	13	14	15	16	17
Data					
PIDConstsD		PIDConstsDFC		PIDDerivFilterON	

For structure see SET message above.

MGMSG_PZ_SET_PPC_NOTCHPARAMS 0x0693
MGMSG_PZ_REQ_PPC_NOTCHPARAMS 0x0694
MGMSG_PZ_GET_PPC_NOTCHPARAMS 0x0695

THIS MESSAGE IS APPLICABLE ONLY TO PPC001 AND PPC102 UNITS

Function:

Due to their construction, most actuators are prone to mechanical resonance at well-defined frequencies. The underlying reason is that all spring-mass systems are natural harmonic oscillators. This proneness to resonance can be a problem in closed loop systems because, coupled with the effect of the feedback, it can result in oscillations. With some actuators, the resonance peak is either weak enough or at a high enough frequency for the resonance not to be troublesome. With other actuators the resonance peak is very significant and needs to be eliminated for operation in a stable closed loop system. The notch filter is an adjustable electronic anti-resonance that can be used to counteract the natural resonance of the mechanical system.
As the resonant frequency of actuators varies with load in addition to the minor variations from product to product, the notch filter is tuneable so that its characteristics can be adjusted to match those of the actuator. In addition to its centre frequency, the bandwidth of the notch (or the equivalent quality factor, often referred to as the Q-factor) can also be adjusted. In simple terms, the Q factor is the centre frequency/bandwidth, and defines how wide the notch is, a higher Q factor defining a narrower ("higher quality") notch. Optimizing the Q factor requires some experimentation but in general a value of 5 to 10 is in most cases a good starting point.

SET:

Command structure (22 bytes)
6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
93	06	10	00	d|	s	Chan Ident		FilterNo		Filter1FC	

12	13	14	15	16	17	18	19	20	21
Data									
Filter1Q		NotchFilter1ON		Filter2FC		Filter2Q		NotchFilter2ON	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
FilterNo	The filter number being addressed… Filter 1 = 1 Filter 2 = 2 Both = 3	word
Filter1FC	The centre frequency of notch filter 1 in the range 20 to 500.	Float
Filter1Q	The Q Factor of Notch Filter 1, in the range 0.2 to 100	Float

NotchFilter1ON	Enables and disables notch filter 1. 1 = ON 2 = OFF	word
Filter2FC	The centre frequency of notch filter 2 in the range 20 to 500.	Float
Filter2Q	The Q Factor of Notch Filter 1, in the range 0.2 to 100	Float
NotchFilter2ON	Enables and disables notch filter 2. 1 = ON 2 = OFF	word

Example: Set the PID constants

TX 93, 06, 10, 00, D0, 01, 01, 00, 01, 00, 96, 00, 32, 00, 01, 00, 00, 00, 00, 00, 00, 00

Header: 90, 06, 0C, 00, D0, 01: SetNotchParams, 16 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
FilterNo: *01, 00:* Address Filter No 1
Filter1FC: 96, 00 Set the centre frequency o0f Filter 1 to 150 Hz
Filter1Q: *32, 00* Set the Q factor of Filter 1 to 50
NotchFilter1ON: *01, 00* Set Notch Filter 1 ON
Filter2FC: *00, 00*
Filter2Q: *00, 00*
NotchFilter2ON: *00, 00*

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
94	06	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
95	06	10	00	d|	s	Chan Ident		FilterNo		Filter1FC	

12	13	14	15	16	17	18	19	20	21
Data									
Filter1Q		NotchFilter1ON		Filter2FC		Filter2Q		NotchFilter2ON	

For structure see SET message above.

MGMSG_PZ_SET_PPC_IOSETTINGS 0x0696
MGMSG_PZ_REQ_PPC_IOSETTINGS 0x0697
MGMSG_PZ_GET_PPC_IOSETTINGS 0x0698

THIS MESSAGE IS APPLICABLE ONLY TO PPC001 AND PPC102 UNITS

Function: This message is used to set various input and output parameter values associated with the rear panel BNC IO connectors.

SET:

Command structure (20 bytes)

6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
96	06	0E	00	d|	s	Chan Ident		ControlSrc		MonitorOPSig	

12	13	14	15	16	17	18	19
Data							
MonitorOPBW		FeedbackSrc		FPBrightness		Reserved	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
ControlSrc	Determines the input source(s) which controls the output from the HV amplifier circuit (i.e. the drive to the piezo actuators) as follows: Software Only = 0 EXT BNC + Software = 1 Joystick + Software = 2 EXT BNC + Joystick + Software = 3 If Software Only (0) is selected, the unit responds only to software inputs and the output to the piezo actuator is that set using the SetVoltOutput method, or the Output knob on the GUI panel. If EXT BNC + Software (1) is selected, the unit sums the analog signal on the rear panel EXT IN BNC connector, with the voltage set using the SetVoltOutput method or the Output knob on the GUI panel. If Joystick + Software (2) is selected, the unit sums the analog signal the external joystick, with the voltage set using the SetVoltOutput method or the Output knob on the GUI panel. If EXT BNC + Joystick + Software (3) is selected, the unit sums all three signals.	word

MonitorOPSig	The signal on the rear panel EXT OUT BNC can be used to monitor the piezo actuator on an oscilloscope or other device. The type of signal can be set as follows: Drive Voltage = 1 Raw Position = 2 Linearized Position = 3 If *Drive Voltage (1)* is selected, the signal driving the EXT OUT (Monitor) BNC is a scaled down version of the piezo output voltage, with 150 V piezo voltage corresponding to 10V. If *Raw Position (2)* is selected, the signal driving the EXT OUT (Monitor) BNC is the output voltage of the position demodulator. This signal shows a slight nonlinearity as a function of position and a small offset voltage. As a result it is not as accurate as the linearized position. However, having not undergone any digital processing it is free of any potential digital signal processing effects and can be more advantageous for loop tuning and transient response measurement. If *Linearized Position (3)* is selected, the signal driving EXT OUT is linearized and scaled so that the 0 to full range corresponds to 0 to 10 Volts.	word
MonitorOPBW	The signal on the rear panel EXT OUT BNC can also be filtered to limit the output bandwidth to the range of interest in most closed loop applications, i.e. 200Hz. The filter is set as follows: No Filter = 1 200 Hz Low Pass Filter = 2	Word
FeedbackSrc	When operating in closed loop mode, the feedback can be supplied by either a Capacitive or a Strain Gauge sensor. This parameter is used to specify the feedback type as follows: Strain Gauge = 1 Capacitive = 2	Word
FPBrightness	The brightness of the LEDs on the front panel of the unit can be set to Bright, Dim or Off as follows: Bright = 1 Dim = 2 Off = 3	word
Reserved	Reserved	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
97	06	01	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
98	06	0E	00	d|	s	Chan Ident		ControlSrc		MonitorOPSig	

12	13	14	15	16	17	18	19
Data							
MonitorOPBW		FeedbackSrc		FPBrightness		Reserved	

See SET message for structure.

MGMSG_PZ_SET_OUTPUTLUT 0x0700
MGMSG_PZ_REQ_OUTPUTLUT 0x0701
MGMSG_PZ_GET_OUTPUTLUT 0x0702

Function:

It is possible to use the controller in an arbitrary Waveform Generator Mode (WGM). Rather than the unit outputting an adjustable but static voltage or position, the WGM allows the user to define a voltage or position sequence to be output, either periodically or a fixed number of times, with a selectable interval between adjacent samples.

This waveform generation function is particularly useful for operations such as scanning over a particular area, or in any other application that requires a predefined movement sequence.

The waveform is stored as values in an array, with a maximum of 8000 samples per channel. The samples can have the meaning of voltage or position; if open loop operation is specified when the samples are output, then their meaning is voltage and vice versa, if the channel is set to closed loop operation, the samples are interpreted as position values. If the waveform to be output requires less than 8000 samples, it is sufficient to download the desired number of samples.

This function is used to load the LUT array with the required output waveform. The applicable channel is specified by the Chan Ident parameter

If only a sub set of the array is being used (as specified by the cyclelength parameter of the SetOutputLUTParams function), then only the first cyclelength values need to be set. In this manner, any arbitrary voltage waveform can be programmed into the LUT.

Note. The LUT values are output by the system at a maximum bandwidth of 7KHz, e.g.500 LUT values will take approximately 71 ms to be clocked out and the full 8000 LUT values will take approximately 1.14 secs.

SET:

Command structure (12 bytes)

6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
00	07	06	00	d|	s	Chan Ident		Index		Output	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Index	The position in the array of the value to be set (0 to 7999 for BPC, 0 to 512 for TPZ).	word
Output	The voltage value to be set. Values are set in the range -32768 to 32767 which corresponds to -100% to 100% of the max HV output (piezo drive voltage).	short

Example: Set output LUT value of 10V (for 150V piezo) in array position 2.

TX 00, 07, 06, 00, D0, 01, 01, 00, 02, 00, 88, 08

Header: 00, 07, 06, 00, D0, 01: SETOUTPUTLUT, 06 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
Index: *02, 00:* sets the value of array position 2
IntConst: *88, 08:* sets the value to 10V. (i.e. 150/10=15, 32767/15=2184, 2184=0888H)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	07	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
02	07	06	00	d|	s	Chan Ident		Index		Output	

For structure see SET message above.

MGMSG_PZ_SET_OUTPUTLUTPARAMS 0x0703
MGMSG_PZ_REQ_OUTPUTLUTPARAMS 0x0704
MGMSG_PZ_GET_OUTPUTLUTPARAMS 0x0705

Function: It is possible to use the controller in an arbitrary Waveform Generator Mode (WGM). Rather than the unit outputting an adjustable but static voltage or position, the WGM allows the user to define a voltage or position sequence to be output, either periodically or a fixed number of times, with a selectable interval between adjacent samples.

This waveform generation function is particularly useful for operations such as scanning over a particular area, or in any other application that requires a predefined movement sequence.

This function is used to set parameters which control the output of the LUT array.

SET:

Command structure (36 bytes)

6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
03	07	1E	00	d|	s	Chan Ident		Mode		CycleLength	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
NumCycles				DelayTime				PreCycleRest			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
PostCycleRest				OPTrigStart		OPTrigWidth				TrigRepCycle	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Mode	Specifies the output mode of the LUT waveform as follows. Values can be 'bitwise or'd together as required. 0x01 - OUTPUTLUT_CONTINUOUS – The waveform is output continuously (i.e. until a StopOPLUT command is received). 0x02 - OUTPUTLUT_FIXED – A fixed number of waveform cycles are output (as specified in the NumCycles parameter). The following values are not applicable to the TPZ001 unit because it has no triggering functionality. 0x04 - OUTPUTLUT_OUTPUTTRIG – Enables Output Triggering. With OP Triggering enabled, the system can be configured to generate one or more hardware trigger pulses during a LUT (waveform) cycle output, as specified in the OPTrigStart parameter below.	word

	0x08 - OUTPUTLUT_INPUTTRIG –Enables Input Triggering. With INPUTTRIG set to 'False', the waveform generator will start as soon as it receives a StartOPLUT command. If however, INPUTTRIG is set to 'True, waveform generation will only start if a software command is received AND the trigger input is in its active state. In most cases, the trigger input will be used to synchronize waveform generation to an external event. In this case, the StartOPLUT command can be viewed as a command to "arm" the waveform generator and the waveform will start as soon as the input becomes active. The trigger input can be used to trigger a single channel or multiple channels. In this latter case ensure that input triggering is enabled on all the desired channels. Using the trigger input for multiple channels is particularly useful to synchronize all channels to the same event. 0x10 - OUTPUTLUT_OUTPUTTRIG_SENSE_HI – determines the voltage sense and edge of the O/P trigger. If this bit is set, the units responds to a rising edge (OV to 5V) trigger. If not set it responds to a falling edge (5V to 0V). 0x20 - OUTPUTLUT_INPUTTRIG_SENSE_HI – determines the voltage sense and edge of the I/P trigger. If this bit is set, the units responds to a rising edge (OV to 5V) trigger. If not set it responds to a falling edge (5V to 0V). 0x40 - OUTPUTLUT_LUTGATED – If set to '1' the trigger acts as a gate, if set to '0' acts as trigger. 0x80 - OUTPUTLUT_OUTPUTTRIG_REPEAT – This parameter is a flag which determines if repeated O/P triggering is enabled. If set, the output trigger is repeated by the interval set in the TrigRepeatCycle parameter. This is useful for multiple triggering during a single voltage O/P sweep.	
CycleLength	Specifies how many samples will be output in each cycle of the waveform. It can be set in the range 0 to 7999 for BPC and MPZ units, and 0 to 512 for TPZ units. It must be less than or equal to the total number of samples that were loaded. (To set the LUT array values for a particular channel, see the SetOutputLUT function).	word
NumCycles	Specifies the number of cycles (1 to 2147483648) to be output when the Mode parameter is set to fixed. If Mode is set to Continuous, the NumCycles parameter is ignored. In both cases, the waveform is not output until a StartOPLUT command is received.	long
DelayTime	Specifies the delay (in sample intervals) that the system waits after setting each LUT output value. By default, the time the system takes to output LUT values (sampling interval) is set at the maximum bandwidth possible, i.e. 7KHz (0.14 ms) for MPZ models, 3.33kHz(0.33 ms) for BPC303 and 4 kHz (0.25 ms) for TPZ units. The DelayTime parameter specifies the time interval between neighbouring samples, i.e. for how long the	long

	sample will remain at its present value. To increase the time between samples, set the DelayTime parameter to the required additional delay (1 to 2147483648 sample intervals). In this way, the user can stretch or shrink the waveform without affecting its overall shape.	
PreCycleRest	In some applications, during waveform generation the first and the last samples may need to be handled differently from the rest of the waveform. For example, in a positioning system it may be necessary to start the movement by staying at a certain position for a specified length of time, then perform a movement, then remain at the last position for another specified length of time. This is the purpose of PreCycleRest and PostCycleRest parameters, i.e. they specify the length of time that the first and last samples are output for, independently of the DelayTime parameter. The PreCycleRest parameter allows a delay time to be set before the system starts to clock out the LUT values. The delay can be set between 0 and 2147483648 sample intervals. The system then outputs the first value in the LUT until the PreCycleRest time has expired.	long
PostCycleRest	In a similar way to PreCycleRest, the PostCycleRest parameter specifies the delay imposed by the system after a LUT table has been output. The delay can be set between 0 and 2147483648 sample intervals. The system then outputs the last value in the cycle until the PostCycleRest time has expired.	long
OPTrigStart	Output triggering is enabled by setting the value 0x04 in the MODE parameter. With Op Triggering enabled, the system can be configured to generate one or more hardware trigger pulses during a LUT (waveform) cycle output. The OPTrigStart parameter specifies the LUT value (position in the LUT array) at which to initiate an output trigger. In this way, it is possible to synchronize an output trigger with the output of a particular voltage value. Values are set in the range 1 to 8000 but must also be less than the CycleLength parameter.	word
OPTrigWidth	sets the width of the output trigger. Values are entered in 1ms increments for BPC20x models.	long
TrigRepeatCycle	specifies the repeat interval between O/P triggers when OUTPUTTRIG_REPEAT is set to True. This parameter is specified in the number of LUT values between triggers (0 to 7999 for MPZ and BPC units, 0 to 512 for TPZ units). If this value is greater than the lCycleLength parameter (set in the SetOPLUTParams method) then by definition, a repeated trigger will not occur during a single waveform cycle output.	word

Example: Set output LUT parameters as follows:
Channel: 1
Mode: OUTPUTLUT continuous
CycleLength: 40
NumCycles: 20
DelayTime: 10
PreCycleRest: 10
PostCycleRest: 10
OPTrigStart: 0
OPTrigWidth: 1
TrigRepeatCycle: 100
TX 03, 07, 1E, 00, D0, 01, 01, 00, 01, 00, 28, 00, 14, 00, 00, 00, 0A, 00, 00, 00, 0A, 00, 00, 00, 0A, 00, 00, 00, 00, 00, 01, 00, 00, 00, 64, 00

Header: 03, 07, 06, 00, D0, 01: SETOUTPUTLUTPARAMS, 30 byte data packet, Generic USB Device.
Channel: 1
Mode: OUTPUTLUT continuous
CycleLength: 00, 28
NumCycles: 00, 00, 00, 14
DelayTime: 00, 00, 00, 0A
PreCycleRest: 00, 00, 00, 0A
PostCycleRest: 00, 00, 00, 0A
OPTrigStart: 00, 00
OPTrigWidth: 00, 00, 00, 01
TrigRepeatCycle: 00, 64

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
04	07	Chan Ident	00	d	s

GET:
Response structure (36 bytes)
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
03	07	1E	00	d|	s	Chan Ident		Mode		CycleLength	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
NumCycles				DelayTime				PreCycleRest			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
PostCycleRest				OPTrigStart		OPTrigWidth				TrigRepCycle	

For structure see SET message above.

MGMSG_PZ_START_LUTOUTPUT 0x0706

Function: This function is used to start the voltage waveform (LUT) outputs. Note. If the IPTrig flag of the SetOPLUTTrigParams function is set to false, this method initiates the waveform immediately. If the IPTrig flag is set to true, then this method 'arms' the system, in readiness for receipt of an input trigger.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
06	07	Chan Ident	00	d	s

MGMSG_PZ_STOP_LUTOUTPUT 0x0707

Function: This function is used to stop the voltage waveform (LUT) outputs.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
07	07	Chan Ident	00	d	s

MGMSG_PZ_SET_EEPROMPARAMS 0x07D0

Function: Used to save the parameter settings for the specified message. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the 'Settings' button found in the lower right hand corner of the user interface).

SET:

Command structure (10 bytes)

6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
D0	07	04	00	d|	s	Chan Ident		MsgID	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
MsgID	The message ID of the message containing the parameters to be saved.	word

Example:

TX D0, 07, 04, 00, D0, 01, 01, 00, 03, 07,

Header: D0, 07, 04, 00, D0, 01: Set_EEPROMPARAMS, 04 byte data packet, Generic USB Device.

Chan Ident: 01, 00: Channel 1

MsgID: Save parameters specified by message 0703 (SetOutputLUTParams).

MGMSG_PZ_SET_TPZ_DISPSETTINGS 0x07D1
MGMSG_PZ_REQ_TPZ_DISPSETTINGS 0x07D2
MGMSG_PZ_GET_TPZ_DISPSETTINGS 0x07D3

Function: Used to set the intensity of the LED display on the front of the TPZ unit.

SET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
D1	07	02	00	d|	s	DispIntensity	

Data Structure:

field	description	format
DispIntensity	The intensity is set as a value from 0 (Off) to 255 (brightest).	word

Example: Set the input source to software and potentiometer.

TX D1, 07, 02, 00, D0, 01, 64, 00,

Header: D1, 07, 02, 00, D0, 01: Set_DISPSETTINGS, 02 byte data packet, Generic USB Device.
DispIntensity: 64, 00: Sets the display brightness to 100 (40%)

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
D2	07	01	00	d	s

Example: Request the display intensity

TX D2, 07, 01, 00, 50, 01

GET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
D3	07	02	00	d|	s	DispIntensity	

See SET for data structure.

MGMSG_PZ_SET_TPZ_IOSETTINGS 0x07D4
MGMSG_PZ_REQ_TPZ_IOSETTINGS 0x07D5
MGMSG_PZ_GET_TPZ_IOSETTINGS 0x07D6

Function: This function is used to set various I/O settings as described below. The settings can be saved (persisted) to the EEPROM by calling the MGMSG_PZ_SET_EEPROMPARAMS function.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D4	07	0A	00	d|	s	Chan Ident		VoltageLimit		HubAnalogIP	

12	13	14	15
Data			
Future Use		Future Use	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
VoltageLimit	The piezo actuator connected to the T-Cube has a specific maximum operating voltage range. This parameter sets the maximum output to the value specified as follows: 0x01 VOLTAGELIMIT_75V 75V limit 0x02 VOLTAGELIMIT_100V 100V limit 0x03 VOLTAGELIMIT_150V 150V limit	word
HubAnalogInput	When the T-Cube Piezo Driver unit is used in conjunction with the T-Cube Strain Gauge Reader (TSG001) on the T-Cube Controller Hub (TCH001), a feedback signal can be passed from the Strain Gauge Reader to the Piezo unit. High precision closed loop operation is then possible using our complete range of feedback-equipped piezo actuators. This parameter is used to select the way in which the feedback signal is routed to the Piezo unit as follows: 0x01 HUB_ANALOGUEIN_A the feedback signals run through all T-Cube bays. 0x02 HUB_ANALOGUEIN_B the feedback signals run between adjacent pairs of T-Cube bays (i.e. 1&2, 3&4, 5&6). This setting is useful when several pairs of Strain Gauge/Piezo Driver cubes are being used on the same hub. 0x03 EXTSIG_SMA the feedback signals run through the rear panel SMA connectors.	word

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
D5	07	01	00	d	s

GET:

Response structure (16 bytes)

6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D4	07	0A	00	d|	s	Chan Ident		VoltageLimit		HubAnalogIP	

12	13	14	15
Data			
Future Use		Future Us	

See SET message for structure.

MGMSG_PZ_SET_ZERO 0x0658

Function: This function applies a voltage of zero volts to the actuator associated with the channel specified by the lChanID parameter, and then reads the position. This reading is then taken to be the zero reference for all subsequent position readings. This routine is typically called during the initialisation or re-initialisation of the piezo arrangement.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
58	06	Chan Ident	00	d	s

MGMSG_PZ_REQ_MAXTRAVEL 0x0650
MGMSG_PZ_GET_MAXTRAVEL 0x0651

Function: In the case of actuators with built in position sensing, the Piezoelectric Control Unit can detect the range of travel of the actuator since this information is programmed in the electronic circuit inside the actuator. This function retrieves the maximum travel for the piezo actuator associated with the channel specified by the Chan Ident parameter, and returns a value (in microns) in the Travel parameter.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
50	06	01	00	d	s

Example: Request the max travel of the actuator associated with Channel 1, bay 2 (0x22)

TX 50, 06, 01, 00, 22, 01

GET:
Response structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
51	06	04	00	d|	s	Chan ID		Travel	

Data Structure:

field	description	format
Chan Ident	The channel being addressed.	word
Travel	The max travel of the actuator associated with the specified channel in the range 0 to 65535 (0 to FFFF). The travel is read from a calibration resistor and is returned in real world units, steps of 100nm.	

Example: Get the maximum travel.

TX 51, 06, 04, 00, 01, A2, 01, 00, C8, 00

Header: 51, 06, 04, 00, A2, 01: Get_Max Travel, 04 byte data packet, d=A2 (i.e. 22 ORed with 80), s=01 (PC).
Channel 1*: 01, 00*:
Travel: 00C8 (200 i.e. 20 µm)

MGMSG_PZ_SET_IOSETTINGS 0x0670
MGMSG_PZ_REQ_IOSETTINGS 0x0671
MGMSG_PZ_GET_IOSETTINGS 0x0672

Function: This function is used to set various I/O settings as described below. The settings can be saved (persisted) to the EEPROM by calling the MGMSG_PZ_SET_EEPROMPARAMS function.

SET:

Command structure (16 bytes)

6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
70	06	0A	00	d|	s	Chan Ident		AmpCurrentLim		AmpLPFilter	

12	13	14	15
Data			
FeedbackSig		BNCTrigORLVOut	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
AmpCurrentLim	This parameter sets the maximum current output for the HV amplifier circuit as follows: CURRENTLIMIT_100MA 0x00 CURRENTLIMIT_250MA 0x01 CURRENTLIMIT_500MA 0x02	word
AmpLPFilter	This parameter sets the value of the hardware low pass filter applied to the HV amplifier output channels. It can be used to improve stability and reduce noise on the HV outputs. It is not channel specific and the Chan Ident parameter is ignored for this particular setting. Values are set as follows: OUTPUTLPFILTER_10HZ 0x00 OUTPUTLPFILTER_100HZ 0x01 OUTPUTLPFILTER_5KHZ 0x02 OUTPUTLPFILTER_NONE 0x03	word
FeedbackSig	For future use. The feedback signal type is locked at AC (strain gauge) and cannot be changed at this time.	
BNCTrigORLVOut	The Control IO BNC connectors on the rear panel are dual function. When set to Low Voltage (LV) outputs they mirror the voltage on the Piezo drive HV connectors and can be connected to an oscilloscope for monitoring purposes. When set to Trigger mode they provide the trigger input and output connections. This function is used to set the mode of the rear panel BNC connectors as follows: BNCMODE_TRIG Trigger Output 0x0000 BNCMODE_LVOUT LV Output 0xFFFF	

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	06	01	00	d	s

GET:
Response structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	06	0A	00	d|	s	Chan Ident		AmpCurrentLim		AmpLPFilter	

12	13	14	15
Data			
FeedbackSig		BNCTrigORLVOut	

See SET message for structure.

MGMSG_PZ_SET_OUTPUTMAXVOLTS 0x0680
MGMSG_PZ_REQ_OUTPUTMAXVOLTS 0x0681
MGMSG_PZ_GET_OUTPUTMAXVOLTS 0x0682

Function: The piezo actuator connected to the unit has a specific maximum operating voltage range: 75, 100 or 150 V. This function sets the maximum voltage for the piezo actuator associated with the specified channel.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
80	06	06	00	d|	s	Chan Ident		Voltage		Flags	

Data Structure:

field	description	format
Chan Ident	The channel being addressed.	word
Voltage	This parameter sets the maximum output to the value specified, in 1/10 volt steps between 0 and 1500 (i.e. 0 to 150 V).	word
Flags	These flags tell the Thorlabs server certain parameters relating to the stage and controller combination. They are not relevant to the SET command and are only used in the GET_OUTPUTMAXVOLTS message	word

Note. When the SET_OUTPUTMAXVOLTS message is sent, a GET_OUTPUTMAXVOLTS message is automatically returned. This is to inform the server that the max output voltage has changed. Similarly, a GET_MAXTRAVEL message is also returned to tell the server the new max travel value.

Example: Set the max output voltage to 100V.

TX 80, 06, 06, 00, D0, 01, 01, 00, E8, 03, 08, 00

Header: 80, 06, 06, 00, D0, 01: Set_OutputMaxVolts, 06 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
Voltage: 03E8 (1000 i.e. 100V)
Flags: N/A

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
81	06	01	00	d	s

GET:

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
82	06	06	00	d|	s	Chan Ident		Voltage		Flags	

Data Structure:

field	description	format
Chan Ident	The channel being addressed.	word
Voltage	This parameter sets the maximum output to the value specified,either 750, 1000 or 1500 (i.e. 75, 100 or 150 V).	word
Flags	These flags tell the Thorlabs server certain parameters relating to the stage and controller combination. The meaning of the individual bits (flags) of the 16 bit integer value is as follows: 0x01 For Future Use 0x02 VOLTAGELIMIT_75V 75V limit 0x04 VOLTAGELIMIT_100V 100V limit 0x05 VOLTAGELIMIT_150V 150V limit	word

Example: Set the max output voltage to 100V.

TX 82, 06, 06, 00, D0, 01, 01, 00, E8, 03, 08, 00

Header: 80, 06, 06, 00, D0, 01: Get_MaxOutputVolts, 06 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
Voltage: 03E8 (1000 i.e. 100V)
Flags: 08, 00: 150 V max voltage

MGMSG_PZ_SET_TPZ_SLEWRATES 0x0683
MGMSG_PZ_REQ_TPZ_SLEWRATES 0x0684
MGMSG_PZ_GET_TPZ_SLEWRATES 0x0685

Function: When stages with delicate internal mechanisms are being driven, it is possible that sudden large changes to the drive voltage could cause damage. This function is used to limit the rate of change of the drive voltage. Different limits may be set for open loop and closed loop operating modes.

Note. The controller is loaded at the factory with default values suitable for driving legacy piezo stages. For newer generation stages, the slew rate is read in automatically. Consequently, these parameters should not require adjustment under normal operating conditions.

SET:

6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
83	06	06	00	d|	s	Chan Ident		SlewOpen		SlewClosed	

Data Structure:

field	description	format
Chan Ident	The channel being addressed.	word
SlewOpen	This parameter sets the maximum slew rate when operating in open loop mode. Values are set in the range 0 to 32767, where 0 disables the limit, and 1 is the slowest rate. Values are calculated in V/ms as follows: $$\text{Slew Rate} = \frac{\text{Value x Max Voltage (i.e. 75, 100 or 150 V)}}{19000}$$	word
SlewClosed	This parameter sets the maximum slew rate when operating in closed loop mode. Values are calculated as above	word

Example: Set the open and closed max slew rates to 10V/ms for a 150V piezo.

TX 83, 06, 06, 00, D0, 01, 01, 00, F2, 04, F2, 04

Header: 80, 06, 06, 00, D0, 01: Set_SlewRates, 06 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
SlewOpen: F2, 04 (10V/ms i.e. 1266 x 150 / 19000)
SlewClosed: F2, 04

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
84	06	01	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
85	06	06	00	d|	s	Chan Ident		SlewOpen		SlewClosed	

See SET message for structure.

MGMSG_PZ_SET_LUTVALUETYPE: 0x0708

Function: It is possible to use the controller in an arbitrary Waveform Generator Mode (WGM). Rather than the unit outputting an adjustable but static voltage or position, the WGM allows the user to define a voltage or position sequence to be output, either periodically or a fixed number of times, with a selectable interval between adjacent samples. This waveform generation function is particularly useful for operations such as scanning over a particular area, or in any other application that requires a predefined movement sequence.

The waveform is stored as values in an array, with a maximum of 8000 samples per channel. The samples can have the meaning of voltage or position; if open loop operation is specified when the samples are output, then their meaning is voltage and vice versa, if the channel is set to closed loop operation, the samples are interpreted as position values. If the waveform to be output requires less than 8000 samples, it is sufficient to download the desired number of samples.

This message specifies whether the samples output from the LUT are voltage or position values.

TX structure (6 bytes):

0	1	2	3	4	5
header only					
08	07	LUTType	00	d	s

Data Structure:

field	description	format
LUTType	The LUT value type: 0x01 LUT values are Voltage 0x02 LUT values are position	char

Example: Set the LUT value type to Volts.

TX, 08,07,01,00,50,01

Notes on using this message.

This method must be called BEFORE the LUT values are downloaded.

The LUT values are scaled to either voltage or position while the LUT is being downloaded. If the value type needs to be changed during operation (e.g. the system was in open loop with volts type selected, but now needs to change to closed loop with position type) the message must be called again, and the LUT values downloaded again.

MGMSG_KPZ_SET_KCUBEMMIPARAMS 0x07F0
MGMSG_KPZ_REQ_KCUBEMMIPARAMS 0x07F1
MGMSG_KPZ_GET_KCUBEMMIPARAMS 0x07F2

This message is applicable only to KPZ101 units

Function: This message is used to configure the operating parameters of the top panel wheel (Joystick) and the display.

SET

Command structure (40 bytes)

6 byte header followed by 34 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F0	07	22	00	d	s	Channel		JSMode		JSVoltGearbox	

12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data													
JSVoltStep				DirSense		PresetVolt1				PresetVolt2			

26	27	28	29	30	31	32	33	34	35	36	37	38	39
Data													
DispBrightness		DispTimeout		DispDimLevel		Reserved		Reserved		Reserved		Reserved	

Data Structure:

field	description	format
Channel	The channel being addressed is always P_MOD_CHAN1 (0x01) encoded as a 16-bit word (0x01 0x00)	word
JSMode	This parameter specifies the operating mode of the wheel/joy stick as follows: 0x01 Voltage Mode - Deflecting the wheel changes the drive voltage. The change is proportional to the deflection. The rate of change is set in the JSVoltGearbox parameter that follows. 0x02 Jog Mode - Deflecting the wheel initiates a jog move, using the parameters specified by the JSVoltStep parameter. One jog step per click of the wheel. 0x03 Go To Voltage Mode - Deflecting the wheel starts a move from the current position to one of the two predefined “teach” positions. The teach positions are specified as a drive voltage in the PresetVolt1 and PresetVolt2 parameters.	word
JSVoltGearbox	The rate of change of voltage, when the JSMode parameter is set to Voltage Adjust Mode. 0x01 - Voltage adjusts at a high rate, i.e. 10 steps per click 0x02 - Voltage adjusts at a medium rate, i.e. 5 steps per click 0x03 - Voltage adjusts at a low rate, i.e. 1 step per click	word
JSVoltStep	The voltage step size when JSMode is set to Jog Mode.	long

DirSense	This parameter specifies the direction of a move initiated by the velocity wheel as follows: 0 Wheel disabled. 1 Upwards rotation of the wheel results in an increased voltage. 2 Upwards rotation of the wheel results in a decreased voltage.	word
PresetVolt1	The preset voltage 1 when operating in Go to Voltage mode.	long
PresetVolt2	The preset voltage 2 when operating in Go to Voltage mode.	long
DispBrightness	In certain applications, it may be necessary to adjust the brightness of the LED display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again.	word
DispTimeout	'Burn In' of the display can occur if it remains static for a long time. To prevent this, the display is automatically dimmed after the time interval specified in the DispTimeout parameter has elapsed. Set in minutes in the range 0 (never dimmed) to 480. The dim level is set in the DispDimLevel parameter below.	word
DispDimLevel	The dim level, as a value from 0 (Off) to 10 (brightest) but is also limited by the DispBrightness parameter.	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
F1	07	Channel	00	d	s

Example: Request the settings for the top panel wheel

TX F1, 07, 01, 00, 50, 01

GET:
Response structure (32 bytes):

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F2	07	22	00	d	s	Channel		JSMode		JSVoltGearbox	

12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data													
JSVoltStep				DirSense		PresetVolt1				PresetVolt2			

26	27	28	29	30	31	32	33	34	35	36	37	38	39
Data													
DispBrightness		DispTimeout		DispDimLevel		Reserved		Reserved		Reserved		Reserved	

For structure see SET message above.

MGMSG_KPZ_SET_KCUBETRIGIOCONFIG 0x07F3
MGMSG_KPZ_REQ_KCUBETRIGIOCONFIG 0x07F4
MGMSG_KPZ_GET_ KCUBETRIGIOCONFIG 0x07F5

Function: The KPZ101 K-Cube piezo controller has two bidirectional trigger ports (TRIG1 and TRIG2) that can be used as a general purpose digital input/output, or can be configured to output a logic level to control external equipment.
When the port is used as an output it provides a push-pull drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuity. The active logic state can be selected High or Low to suit the requirements of the application.

This message sets the operating parameters of the TRIG1 and TRIG2 connectors on the front panel of the unit.

Warning. Do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

Trigger Modes
Input Trigger Modes
When configured as an input, the TRIG ports can be used as a general purpose digital input, or for triggering a drive voltage change as follows:

0x00 The trigger IO is disabled.
0x01 General purpose logic input (read through status bits using the PZ_GET_PZSTATUSUPDATE message).
0x02 Input trigger for voltage step up. On receipt of the trigger, the drive voltage increases by the value set in the SetKCubeMMIParams method, VoltStep parameter.
0x03 Input trigger for voltage step down. On receipt of the trigger, the drive voltage decreases by the value set in the SetKCubeMMIParams method, VoltStep parameter.

When used for triggering a move, the port is edge sensitive. In other words, it has to see a transition from the inactive to the active logic state (Low->High or High->Low) for the trigger input to be recognized. For the same reason a sustained logic level will not trigger repeated moves. The trigger input has to return to its inactive state first in order to start the next trigger.

Output Trigger Modes
When configured as an output, the TRIG ports can be used as a general purpose digital output.

0x0A General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).

Trigger Polarity
The polarity of the trigger pulse is specified in the TrigPolarity parameters as follows:
0x01 The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).
0x02 The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

SET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F3	07	16	00	d|	s	Channel		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21
Data									
Trig2Mode		Trig2Polarity		Reserved		Reserved		Reserved	

22	23	24	25	26	27
Data					
Reserved		Reserved		Reserved	

Data Structure:

field	description	format
Channel	The channel being addressed is always (e.g. 0x01) encoded as a 16-bit word (0x01 0x00)	word
Trig1Mode	TRIG1 operating mode:	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) .	word
Trig2Mode	TRIG2 operating mode:	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low) .	word
Reserved		word

Example: Set the Trigger parameters for KPZ101 as follows:
Trig1Mode – TrigIn_VoltStepUp
Trig1Polarity – High
Trig2Mode – Disabled
Trig2Polarity – N/A

TX F3, 07, 0C, 00, D0, 01, 01, 00, 02, 00, 01, 00, 00, 00, 00, 00, 00, 00

Header: F3, 07, 0C, 00, D0, 01: Set_KCube_TrigIOConfig, 12 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
Trig1Mode – 02, 00 TrigIn_VoltStepUp
Trig1Polarity – 01,00 High
Trig2Mode – 00,00 Disabled
Trig2Polarity – 00,00 N/A

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
F4	07	01	00	d	s

GET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F5	07	16	00	d|	s	Channel		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21
Data									
Trig2Mode		Trig2Polarity		Reserved		Reserved		Reserved	

22	23	24	25	26	27
Data					
Reserved		Reserved		Reserved	

See SET message for structure.

MGMSG_PZ_SET_TSG_IOSETTINGS 0x07DA
MGMSG_PZ_REQ_TSG_IOSETTINGS 0x07DB
MGMSG_PZ_GET_TSG_IOSETTINGS 0x07DC

Function: When the T-Cube Strain Gauge Reader is used in conjunction with the T-Cube Piezo Driver unit (TPZ001) on the T-Cube Controller Hub (TCH001), a feedback signal can be passed from the Strain Gauge Reader to the Piezo unit. High precision closed loop operation is then possible using our complete range of feedback-equipped piezo actuators.
This method is used to select the way in which the feedback signal is routed back to the Piezo unit.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
DA	07	0E	00	d|	s	Chan Ident		HubAnalogOP		DisplayMode	

12	13	14	15	16	17	18	19
Data							
ForceCalib				Future Use		Future Use	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always (e.g. 0x01) encoded as a 16-bit word (0x01 0x00)	word
HubAnalogueOutput	When the T-Cube Strain Gauge Reader is used in conjunction with the T-Cube Piezo Driver unit (TPZ001) on the T-Cube Controller Hub (TCH001), a feedback signal can be passed from the Strain Gauge Reader to the Piezo unit. High precision closed loop operation is then possible using our complete range of feedback-equipped piezo actuators. This message is used to select the way in which the feedback signal is routed back to the Piezo unit If set to 0x01 HUB_ANALOGUEOUT_1, the feedback signals run through all T-Cube bays. If set to 0x02 HUB_ANALOGUEOUT_2,the feedback signals run between adjacent pairs of T-Cube bays (i.e. 1&2, 3&4, 5&6). This setting is useful when several pairs of Strain Gauge/Piezo Driver cubes are being used on the same hub.	word

Display Mode	The LED display window on the front of the unit (and the display on the GUI panel) can be set to display the strain gauge signal as a position (microns), a voltage (Volts) or as a force (Newtons). This parameter sets the display mode as follows If set to 0x01 DISPUNITS_POSITION, the display shows the strain gauge signal as a position in microns. If set to 0x02 DISPUNITS_VOLTAGE, the display shows the strain gauge signal as a voltage. If set to 0x03 DISPUNITS_FORCE, the display shows the strain gauge signal as a force	word
ForceCalib	If using a force sensor with the TSG001 unit, the Force Sensor has a specific maximum operating force. This parameter sets the force calibration factor in steps of 0.001 N between 1 and 1000. The default setting for this parameter is H7530 (30,000), to be compatible with our FSC102 force sensor, which is specified to read forces up to 30N.	word

Example: Set the IO settings as follows.

TX DA, 07, 0E, 00, D0, 01, 01, 00, 01, 00, 02, 00, 30, 75, 00, 00, 00, 00, 00, 00

Header: DA, 07, 0E, 00, D0, 01: Set_TSG_IOSettings, 14 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
HubAnalogueOutput: 01, 00 (Hub Analogue Output A)
Display Mode: 02, 00 (Display Voltage
Force Calibration: 30, 75 30,000 x 0.001 = 30 N

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
DB	07	01	00	d	s

GET:
Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
DC	07	0E	00	d|	s	Chan Ident		HubAnalogOP		DisplayMode	

12	13	14	15	16	17	18	19
Data							
ForceCalib				Future Use		Future Use	

See SET message for structure.

MGMSG_PZ_REQ_TSG_READING 0x07DD
MGMSG_PZ_GET_TSG_READING 0x07DE

Function: This message returns the current reading of the strain gauge The units applicable are dependent on the current operating mode (set using the DisplayMode parameter of the SET_TSG_IOSETTINGS message.

REQUEST:
Command structure (6 bytes)

0	1	2	3	4	5
header only					
DD	07	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
DE	07	06	00	d|	s	Chan Ident		Reading		Smoothed	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
Reading	The current reading of the strain gauge unit. If the unit is operating in Position mode, then the returned value is a position in microns. If the unit is in Voltage mode, then the returned reading is a Voltage. If the controller is in 'Force Sensing Mode' then the parameter returns a force value in Newtons. Values are returned in the range -32767 to 32768, which corresponds to -100% to 100% of the maximum voltage, travel or force. The returned data values are sampled at 500Hz. This is particularly useful in touch probe or force sensing applications where rapid polling of the force reading is important. Display mode and Max Force are described in the MGMSG_PZ_GET_TSG_IOSETTINGS message. Max Travel is described in the MGMSG_PZ_GET_MAXTRAVEL message.	short
Smoothed		word

Example: Get the readings for channel 1.

RX DE, 07, 06, 00, 81, 50, 01, 00, 52, 00, 50, 00,

Header: DE, 07, 06, 00, 81, 50: Get_TSG_Readings, 6 byte data packet, d=D0 (i.e. 01 ORed with 80 i.e. PC), s=50 (Generic USB device).
Channel 1: 01, 00
Reading: 52, 00 (i.e. 82)
Smoothed: 52, 00

MGMSG_KSG_SET_KCUBEMMIPARAMS 0x07F6
MGMSG_KSG_REQ_KCUBEMMIPARAMS 0x07F7
MGMSG_KSG_GET_KCUBEMMIPARAMS 0x07F8

Function: Used to set the intensity of the OLED display on the TOP of the KSG101 unit. Intensity is set as a percentage of full brightness in the range 0 (off) to 100%. Also used to set the display time out and dim level as described below.

SET:

Command structure (14 bytes)

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
F6	07	08	00	d|	s	ChanIdent		DispIntensity		DispTimeout		DispDimLevel	

Data Structure:

field	description	format
ChanIdent	The channel being addressed (i.e. 1)	word
DispIntensity	In certain applications, it may be necessary to adjust the brightness of the LED display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again.	word
DispTimeout	'Burn In' of the display can occur if it remains static for a long time. To prevent this, the display is automatically dimmed after the time interval specified in the DispTimeout parameter has elapsed. Set in minutes in the range 0 (never dimmed) to 480. The dim level is set in the DispDimLevel parameter below.	word
DispDimLevel	The dim level, as a value from 0 (Off) to 10 (brightest) but is also limited by the DispBrightness parameter.	word

Example: Set the Display intensity 50%, the Time out to 5 minutes and the dim level to 20%. .

TX F6, 07, 08, 00, D0, 01, 01, 00, 32, 00

Header: F6, 07, 04, 00, D0, 01: Set_KCUBEMMIPARAMS, 08 byte data packet, Generic USB Device.
ChanIdent: 01, 00: Sets channel 1
DispIntensity: 32, 00: Sets the display brightness to 50%
DispTimeout: 05, 00: Sets the display brightness to 5 minutes
DispDimLevel: 14, 00: Sets the display brightness to 20%

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
F7	07	01	00	d	s

Example: Request the display intensity

TX F6, 07, 01, 00, 50, 01

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
F8	07	08	00	d|	s	ChanIdent		DispIntensity		DispTimeout		DispDimLevel	

See SET for data structure.

MGMSG_KSG_SET_KCUBETRIGIOCONFIG 0x07F9
MGMSG_KSG_REQ_KCUBETRIGIOCONFIG 0x07FA
MGMSG_KSG_GET_ KCUBETRIGIOCONFIG 0x07FB

Function: The KSG101 K-Cube strain gauge reader has two bidirectional trigger ports (TRIG1 and TRIG2) that can be used as a general purpose digital input/output, or can be configured to output a logic level to control external equipment.
When the port is used as an output it provides a push-pull drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuity. The active logic state can be selected High or Low to suit the requirements of the application.

This message sets the operating parameters of the TRIG1 and TRIG2 connectors on the front panel of the unit.

Warning. Do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

The Trigger can be used to monitor a specific area, and output a signal when the device moves away from this region of interest. This signal can then be used to give a warning by sounding a bell or turning on an LED. The triggers are set using a combination of the Trig1Mode and Trig2Mode parameters, and the LowerLim and UpperLim parameters.

Trigger Modes
0x00 - TRIG_DISABLED The trigger IO is disabled
0x01 - TRIGIN_GPI General purpose logic input (read through status bits using the PZ_GET_PZSTATUSUPDATE message).
0x0A - TRIGOUT_GPO General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).
0x0B - TRIG_OUT_LESSTHANLOWERLIMIT The trigger is active when the strain gauge input is less than the lower limit, set in the LowerLim parameter.
0x0C TRIG_OUT_MORETHANLOWERLIMIT - The trigger is active when the strain gauge input is greater than the lower limit.
0x0D TRIG_OUT_LESSTHANUPPERLIMIT - The trigger is active when the strain gauge input is less than the upper limit, set in the UpperLim parameter.
0x0E TRIG_OUT_MORETHANUPPERLIMIT - The trigger is active when the strain gauge input is greater than the upper limit.
0x0F TRIG_OUT_BETWEENLIMITS - The trigger is active when the strain gauge input is between the two limits.
0x10 TRIG_OUT_OUTSIDELIMITS - The trigger is active when the strain gauge input is outside either of the two limits.

Trigger Polarity
The polarity of the trigger pulse is specified in the TrigPolarity parameters as follows:
0x01 The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).
0x02 The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

SET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
F9	07	16	00	d|	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig2Mode		Trig2Polarity		LowerLim				UpperLim			

24	25	26	27
Data			
SmoothingSamples		Reserved	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always (e.g. 0x01) encoded as a 16-bit word (0x01 0x00)	word
Trig1Mode	TRIG1 operating mode:	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) .	word
Trig2Mode	TRIG2 operating mode:	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low) .	word
LowerLim	The lower limit described in the trigger mode details above, set in the range -100 to 100.	Long
UpperLim	The upper limit described in the trigger mode details above, set in the range -100 to 100.	Long
SmoothingSamples	The reading shown on the display is an average of the number of samples set in the SmoothingSamples parameter, between 0 and 1000. As a new sample is taken, the earliest sample is discarded.	word
Reserved		

Example: Set the Trigger parameters for KSG101 as follows:
Trig1Mode – TrigOut_LESSTHANLOWERLIMIT
Trig1Polarity – High
Trig2Mode – Disabled
Trig2Polarity – N/A
LowerLim – Zero
UpperLim – 100
SmoothingSamples - 1000

TX F9, 07, 16, 00, D0, 01, 01, 00, 0B, 00, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00, 64, 00, 00, 00, E3, 08

Header: F9, 07, 16, 00, D0, 01: Set_KCube_TrigIOConfig, 22 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).
Channel 1*: 01, 00*:
Trig1Mode – 0B, 00 TrigOut_LESSTHANLOWERLIMIT
Trig1Polarity – 01,00 High
Trig2Mode – 00,00 Disabled
Trig2Polarity – 00,00 N/A
LowerLim – 00,00,00,00 Zero
UpperLim – 64,00 i.e. 100
SmoothingSamples – E8, 03 i.e. 1000

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
FA	07	01	00	d	s

GET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
FB	07	16	00	d|	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig2Mode		Trig2Polarity		LowerLim				UpperLim			

24	25	26	27
Data			
SmoothingSamples		Reserved	

See SET message for structure.

NanoTrak Control Messages

Introduction

The ‘NanoTrak’ ActiveX Control provides the functionality required for a client application to control one or more NanoTrak auto-alignment controller products. The NanoTrak system comes in benchtop (BNT001), T-Cube (TNA001) and 19" rack modular (MNA601) formats, all of which are covered by the NanoTrak ActiveX Control.
The messages of the NanoTraks object can then be used to perform activities such as latching/unlatching, reading power levels, obtaining/setting circle size and position and determining if 'NanoTracking' is currently taking place.
For details on the use of the NanoTrak controller, and information on the principles of operation, refer to the NanoTrak Operating Guide.

NOTE. The NanoTrak can be set to operate as a piezo amplifier. When operated in this mode, some piezo control messages may also be sent or returned.

MGMSG_PZ_SET_NTMODE 0x0603

Function: The NanoTrak unit can be used as a standard piezo amplifier, or as a NanoTrak Auto-alignment unit. This message sets the unit to piezo operation, or one of the NanoTrak operating modes as described below. The mode of operation is set in byte 2 of the message as follows:

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
03	06	State	00	d	s

Data Structure:

field	description	format
State	01 Sets the unit to Piezo mode. **Note**. The hardware unit must be rebooted before changes to operating mode can take effect. **Note**. When the HW operating mode of a NanoTrak unit has been changed to Piezo operation, then the Piezo ActiveX control must be used to communicate with the unit. Use the same serial number as used on the NanoTrak control in order to establish communication with the unit. 02 Latch mode. In this mode, scanning is disabled and the piezo drives are held at the present position. 03 Track mode. In this mode, the NanoTrak detects any drop in signal strength resulting from misalignment of the input and output devices, and makes vertical and horizontal positional adjustments to maintain the maximum throughput. 04 Horizontal Track mode. In this mode, the NanoTrak detects any drop in signal strength resulting from misalignment of the input and output devices, and makes horizontal positional adjustments to maintain the maximum throughput. 05 Vertical Track mode. In this mode, the NanoTrak detects any drop in signal strength resulting from misalignment of the input and output devices, and makes vertical positional adjustments to maintain the maximum throughput.	short

Example: Set the tracking mode to Latch

TX 03, 06, 02, 00, 50, 01,

MGMSG_PZ_REQ_NTMODE 0x0604
MGMSG_PZ_GET_NTMODE 0x0605

Function: The NanoTrak unit can be used as a standard piezo amplifier, or as a NanoTrak Auto-alignment unit. This message gets the present operating mode of the unit as described below. The mode of operation is returned in byte 2 of the message as follows:

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
04	06	00	00	d	s

GET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
05	06	State	Mode	d	s

Data Structure:

field	description	format
State	The Tracking state 01 NanoTracking off. The unit is in Piezo mode. 02 Latch mode. In this mode, scanning is disabled and the piezo drives are held at the present position. 03 Tracking ON No Signal. In this mode, the NanoTrak is tracking but the signal power is below the threshold power set by the user in the Set_NTTrackThreshold message. 04 Tracking ON, Signal Attained. In this mode, the threshold power has been detected and the NanoTrak is tracking normally.	short
Mode	The Tracking Mode. 01 Dual axis (X and Y) tracking. 02 Horizontal (X) axis tracking. 03 Vertical (Y) axis tracking.	

Example

TX 05, 06, 04, 01, 01, 50
Mode is Tracking Signal (0x04) and dual axis (Both X and Y tracking) (0x01)

MGMSG_PZ_SET_NTTRACKTHRESHOLD 0x0606
MGMSG_PZ_REQ_NTTRACKTHRESHOLD 0x0607
MGMSG_PZ_GET_NTTRACKTHRESHOLD 0x0608

Function: This message sets the tracking threshold of the NanoTrak. The value is set in Amps, and is dependent upon the application. Typically, the value is set to lie above the 'noise floor' of the particular physical arrangement. When the input signal level exceeds this value, the tracking LED is lit on the GUI panel. Note there is no guarantee that tracking is taking place if this threshold value is set inappropriately. E.g. if the tracking threshold is set to below the noise floor, then the GUI will show a lit tracking LED even though no tracking is taking place.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
06	06	04	00	d|	s	ThresholdAbsReading			

Data Structure:

field	description	format
ThresholdAbsReading	The tracking threshold of the NanoTrak. This is the absolute TIA reading (PIN current). The value set in Amps as a 4-byte floating point number in the range 1 x 10^{-9} to 1 x 10^{-3} (i.e. 1 nA to 1 mA).	Float

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
07	06	00	00	d	s

GET:
Command structure (10 bytes):

0	1	2	3	4	5	6	7	8	9
header						*Data*			
08	06	04	00	d|	s	ThresholdAbsReading			

See SET for structure.

MGMSG_PZ_SET_NTCIRCHOMEPOS 0x0609
MGMSG_PZ_REQ_NTCIRCHOMEPOS 0x0610
MGMSG_PZ_GET_NTCIRCHOMEPOS 0x0611

Function: This message sets the circle home position to the horizontal and vertical coordinates specified in the CircHomePosA and CircHomePosB parameters respectively.
The home position is used when the Move_NTCircToHomePos message is called

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
06	06	04	00	d|	s	CircHomePosA		CircHomePosB	

Data Structure:

field	description	format
CircHomePosA	The horizontal co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word
CircHomePosB	The vertical co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word

Example: Set the NanoTrak circle home position to be screen centre.

TX 09 06, 04, 00, D0, 01, FF, 7F, FF, 7F,

Header: 09, 06, 04, 00, D0, 01: Set_NTCircHomePos, 04 byte data packet, Generic USB Device.
CircHomePosA: FF, 7F: Sets the horizontal co-ordinate to 32767 (i.e. 50% of O/P Voltage or 5 NT units)
CircHomePosB: FF, 7F: Sets the vertical co-ordinate to 32767 (i.e. 50% of O/P Voltage or 5 NT units)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
10	06	00	00	d	s

GET:
Command structure (10 bytes):

0	1	2	3	4	5	6	7	8	9
header						*Data*			
11	06	04	00	d|	s	CircHomePosA		CircHomePosB	

See SET for structure.

MGMSG_PZ_MOVE_NTCIRCTOHOMEPOS 0x0612

Function: This message moves the circle to the 'Home' position as set by the Set_NTCircHomePos message

SET:
Command structure (6 bytes)

0	1	2	3	4	5
header					
12	06	00	00	d	s

Example: Move the NanoTrak circle to the home position.

TX, 12, 06, 00, 00, 50, 01,

MGMSG_PZ_REQ_NTCIRCCENTREPOS 0x0613
MGMSG_PZ_GET_NTCIRCCENTREPOS 0x0614

Function: This message obtains the current horizontal and vertical position of the circle, together with other signal and range parameters relating to NanoTrak operation as described below.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	06	01	00	d	s

GET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
06	06	0E	00	d|	s	CircPosA		CircPosB	

10	11	12	13	14	15	16	17	18	19
Data									
AbsReading				RelReading		Range		UnderOverRead	

Data Structure:

field	description	format
CircPosA	The horizontal co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word
CircPosB	The vertical co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word
AbsReading	The absolute TIA (PIN) current or BNC voltage value at the current position. The value is returned as a 4 byte floating point value in the range 1×10^{-9} to 1×10^{-3} (i.e. 1 nA to 1 mA or 1 to 10 V). The input source, TIA or BNC is set in the Set_NTFeedbackSRC message.	float
RelReading	The relative signal strength at the current position, in the range 0 to 32767 (i.e. 0 to 100% of the range currently selected). This value matches the length of the input signal bargraph on the GUI panel. (e.g. if the 3 µA range is currently selected, then a RelReading value of 16384 (50%) equates to 1.5 µA).	word
Range	The NanoTrak unit is equipped with an internal trans-impedance amplifier (TIS) circuit (and associated range/power level displays and control buttons in the GUI). This amplifier operates when an external input signal is connected to the Optical/PIN connector on the rear panel. There are 14 range settings (1 - 14) that can be used to select the best range to measure the input signal (displayed on the GUI panel relative input signal bar and display).	word

	Note. Range 1 and 2 (3 nA and 10 nA) are not applicable to TNA001 T-Cube units. This parameter returns the input signal range currently selected, defined as follows: **Range** **BNT, TNA, MNA** **KNA** **Returned** Range 1 3 nA 5 nA 0x03 Range 2 10 nA 16.6 nA 0x04 Range 3 30 nA 50 nA 0x05 Range 4 100 nA 166 nA 0x06 Range 5 300 nA 500 nA 0x07 Range 6 1 µA 1.65 µA 0x08 Range 7 3 µA 5.0 µA 0x09 Range 8 10 µA 16 µA 0x0A Range 9 30 µA 50 µA 0x0B Range 10 100 µA 166 µA 0x0C Range 11 300 µA 500 µA 0x0D Range 12 1 mA 1.66 m 0x0E Range 13 3 mA 5 mA 0x0F Range 14 10 mA N/A 0x10	
UnderOverRead	This parameter returns a value that identifies whether the unit is under reading or over reading the input signal as follows: 0x01 power signal is within current TIA range 0x02 power signal is under-reading for current TIA 0x03 power signal is over-reading for current TIA range e.g. if a user specified range of 3 µA is currently applied, this parameter returns '0x03' (Over read)' for input signals greater than 3 µA.	word

Example:

RX 14, 06, 0E, 00, 81, 50, 73, 63, 2A, F3, 00, 00, 00, 00, 00, 00, 05, 00, 02, 00

Header: 14, 06, 0E, 00, 81, 50: Get_NTCircCentrePos, 14 byte data packet, Generic USB Device.

CircPosA;	0x6373	25459 (25459/65535 = 39%)
CircPosB;	0xF32A	62250 (62250/65535 = 95%)
AbsReading;	0x00000000	0V
RelReading;	0x0000	0V
Range;	0x0005	Range 3 (i.e. 30 nA)
UnderOverRead;	0x0002	Signal is under reading for range.

MGMSG_PZ_SET_NTCIRCPARAMS 0x0618
MGMSG_PZ_REQ_NTCIRCPARAMS 0x0619
MGMSG_PZ_GET_NTCIRCPARAMS 0x0620

Function: This message obtains sets various scanning circle parameters as described below.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
18	06	0C	00	d|	s	CircDiaMode		CircDiaSW	

10	11	12	13	14	15	16	17
Data							
CircOscFreq		AbsPwrMinCircDia		AbsPwrMaxCircDia		AbsPwrAdjustType	

Data Structure:

field	description	format
CircDiaMode	This parameter allows the different modes of circle diameter adjustment to be enabled and disabled as follows: 0x01 NTCIRCDIA_SW the circle diameter remains at the value set using the CircDiaSW parameter below. 0x02 NTCIRCDIA_ABSPWR the circle diameter is set by absolute power input value (depending on adjustment algorithm selected in the AbsPwrAdjustType parameter - see below) 0x03 NTCIRCDIA_LUT the circle diameter is adjusted automatically, using a table of TIA range dependent values (set using the SetCircDiaLUT message.	word
CircDiaSW	This parameter sets the NT circle diameter if NTCIRCDIA_SW (0x01) is selected in the CircDiaMode parameter above. The diameter is set in the range 0 to 65535, which relates to 0% to 100% output voltage –(i.e. 0 to 10 NT units).	word
CircOscFreq	This parameter contains the number of samples taken in one revolution of the scanning circle and is used to set the scanning frequency of the NanoTrak circle. The circle scanning frequency lies in the range 17.5 Hz to 87.5 Hz for TNA001 and 20 Hz to 190 Hz for the BNT001. The factory default setting for the scanning frequency is 43.75Hz. This means that a stage driven by the NanoTrak makes 43.75 circular movements per second. Different frequency settings allow more than one NanoTrak to be used in the same alignment scenario. The scanning frequency is derived from the NanoTrak sampling frequency of 7000 Hz and the CircOscFreq	word

	value which is calculated as follows: CircOscFreq = 7000 / scanning frequency **Note**. The CircOscFreq parameter must be entered as a multiple of '4'.	
AbsPwrMinCircDia	The minimum circle diameter. Applicable only if the CircDiaMode parameter above is set to NTCIRCDIA_ABSPWR (0x02). The diameter is set in the range 0 to 32767, which relates to 0% to 50% output voltage –(i.e. 0 to 5 NT units).	word
AbsPwrMaxCircDia	The maximum circle diameter. Applicable only if the CircDiaMode parameter above is set to NTCIRCDIA_ABSPWR (0x02). The diameter is set in the range 0 to 32767, which relates to 0% to 50% output voltage –(i.e. 0 to 5 NT units).	word
AbsPwrAdjustType	This parameter sets the adjustment type and is applicable only if CircDiaMode parameter above is set to NTCIRCDIA_ABSPWR (0x02). 0x01 NTABSPWRCIRCADJUST_LIN inverse linear adjustment 0x02 NTABSPWRCIRCADJUST_LOG inverse log adjustment 0x03 NTABSPWRCIRCADJUST_X2 inverse square adjustment 0x04 NTABSPWRCIRCADJUST_X3 inverse cube adjustment	word

Example

TX 18, 06, 0C, 00, D0, 01, 01, 00, 9A, 19, A0, 00, CC, 0C, 99, 19, 01, 00

Header: 18, 06, 0C, 00, D0, 01: Set_NTCircParams, 12 byte data packet, Generic USB Device.
CircDiaMode; 0x0001 Software setting mode
CircDiaSW; 0x199A 6554 6554/65535 = 10% of O/P voltage (1 NT unit)
CircOscFreq; 0x00A0 160 7000/160 = 43.75 Hz
AbsPwrMinCircDia; 0x0CCC 3276 5% or 0.5 NT units
AbsPwrMaxCircDia; 0x1999 6553 10% or 1 NT unit
AbsPwrAdjustType; 0x0001 inverse linear adjust type.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
19	06	01	00	d	s

GET:

Command structure (18 bytes)

6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
20	06	0C	00	d|	s	CircDiaMode		CircDiaSW	

10	11	12	13	14	15	16	17
Data							
CircOscFreq		AbsPwrMinCircDia		AbsPwrMaxCircDia		AbsPwrAdjustType	

See SET for structure

MGMSG_PZ_SET_NTCIRCDIA 0x061A

Function: This message sets the NT circle diameter and can be used as an alternative to the Set_NTCircParams message described previously. The diameter is set in the range 0 to 65535, which relates to 0% to 100% output voltage (i.e. 0 to 10 NT units).

SET:

Command structure (6 bytes)

0	1	2	3	4	5
header					
1A	06	CircDia	00	d	s

Example: Set the NanoTrak circle diameter to 10% (i.e. 1 NT unit).

TX, 1A, 06, 99, 19, 50, 01,

H1999 = 6553 6553/65535 = 10%

MGMSG_PZ_SET_NTCIRCDIALUT 0x0621
MGMSG_PZ_REQ_NTCIRCDIALUT 0x0622
MGMSG_PZ_GET_NTCIRCDIALUT 0x0623

Function: This message enables a look up table (LUT) of circle diameter values to be specified as a function of input range. When automatic LUT diameter adjustment mode is enabled (using the CircDiaMode parameter in the Set_NTCircParams message), the system uses values in this LUT to modify circle diameter in relation to the input range currently selected.
This LUT diameter adjustment mode allows appropriate circle diameters to be applied on an application specific basis.

SET:
Command structure (38 bytes)
6 byte header followed by 32 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
21	06	20	00	d|	s	LUTVal		LUTVal		LUTVal	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal	

24	25	26	27	28	29	30	31	32	33	34	35	36	36
Data													
LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal	

Data Structure:

field	description	format
CircDias	This parameter contains the circle diameter values for each range of the NanoTrak. The values are entered in range order in a 32 byte array. **Note**. On the BNT001 unit bytes 1 through 4 of the array are ignored and Range 1 starts in Byte 5. **Note**. On the TNA001 unit bytes 1 through 8 of the array are ignored and Range 1 starts in Byte 9. The diameters are entered in the range 0 to 65535 (0 to FFFF), which relates to 0% to 100% output voltage (i.e. 0 to 10 NT units).	array

Example: Enter the NanoTrak cirle diameter LUT values.

TX 21, 06, 20, 00, D0, 01, 00, 00, 00, 00, 34, 33, A4, 30, 16, 2E, 86, 2B, F6, 28, 68, 26, D8, 23, 48, 21, B8, 1E, 2A, 1C, 9A, 19, 0A, 17, 7C, 14, EC, 11

Header: 21, 06, 20, 00, D0, 01: Set_NTCircHomePos, 32 byte data packet, Generic USB Device.
CircDias: The various range related LUT values entered in range order)

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
22	06	00	00	d	s

GET:

Command structure (38 bytes)

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
23	06	20	00	d|	s	Not Used		Not Used		LUTVal	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal	

24	25	26	27	28	29	30	31	32	33	34	35	36	36
Data													
LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal		LUTVal	

See SET for structure.

MGMSG_PZ_SET_NTPHASECOMPPARAMS 0x0626
MGMSG_PZ_REQ_NTPHASECOMPPARAMS 0x0627
MGMSG_PZ_GET_NTPHASECOMPPARAMS 0x0628

Function: The feedback loop scenario in a typical NanoTrak application can involve the operation of various electronic and electromechanical components (e.g. power meters and piezo actuators) that could introduce phase shifts around the loop and thereby affect tracking efficiency and stability. These phase shifts can be cancelled by setting the 'Phase Compensation' factors.
This message sets the phase compensation for the horizontal and vertical components of the circle path in the range 0 to 360 degrees. Typically both phase offsets will be set the same, although some electromechanical systems may exhibit different phase lags in the different components of travel and so require different values.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
26	06	06	00	d|	s	PhaseCompMode		PhaseCompASW		PhaseCompBSW	

Data Structure:

field	description	format
PhaseCompMode	Currently, the phase compensation mode is not adjustable, and is locked at manual (software) adjustment.	word
PhaseCompASW	The horizontal axis phase compensation value, entered in real world units and calculated as follows:- value = (phase angle [degrees] / 360) * CircOscFreq See the PZ_SET_NTCIRCPARAMS message for details on the CircOscFreq parameter **Note**. Negative phase values must be made positive by subtraction from 360 before the calculation is made.	short
PhaseCompBSW	The vertical axis phase compensation value, entered in real world units and calculated as follows:- value = (phase angle [degrees] / 360) * CircOscFreq See the PZ_SET_NTCIRCPARAMS message for details on the CircOscFreq parameter **Note**. Negative phase values must be made positive by subtraction from 360 before the calculation is made.	short

Example: Set the NanoTrak circle home position to be screen centre.

TX 26, 06, 06, 00, D0, 01, 02, 00, 93, 00, 93, 00

Header: 26, 06, 06, 00, D0, 01: Set_NTPhaseCompParams, 06 byte data packet, Generic USB Device.

PhaseCompMode; 0x0002 Locked at Software Adjustment mode.

PhaseCompASW; 0x0093 147

Therefore, for circle scanning freq of 44, Phase Angle = 147/(7000/44) x 360 = -30°

PhaseCompBSW 0x0093

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
27	06	00	00	d	s

GET:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
28	06	06	00	d|	s	PhaseCompMode		PhaseCompASW		PhaseCompBSW	

See SET for structure.

MGMSG_PZ_SET_NTTIARANGEPARAMS 0x0630
MGMSG_PZ_REQ_NTTIARANGEPARAMS 0x0631
MGMSG_PZ_GET_NTTIARANGEPARAMS 0x0632

Function: This message is used to select manual (software) or auto ranging, and to modify the ranging characteristics in each case.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
30	06	0C	00	d|	s	RangeMode		RangeUpLimit	

10	11	12	13	14	15	16	17
RangeDownLimit		SettleSamples		RangeChangeType		RangeSW	

Data Structure:

field	description	format
RangeMode	This parameter specifies the ranging mode of the unit as follows: 0x01 RANGE_AUTO change to Auto ranging at the range currently selected 0x02 RANGE_SW change to manual ranging at the range currently selected 0x03 RANGE_SWSET change to manual ranging at the range set in the SetRange method (or the 'Settings' panel) 0x04 RANGE_AUTOSET change to Auto ranging at the range set in the RangeSW parameter below.	word
RangeUpLimit	Only applicable if Auto Ranging is selected in the RangeMode parameter above. This parameter sets the upper range limit as a percentage of the present range, 0 to 1000 = 0 to 100%. When autoranging, the NanoTrak unit adjusts continually the TIA range as appropriate for the input signal level. When the relative signal rises above the limit specified in this parameter, the unit increments the range to the next higher setting. The relative signal is displayed on the NanoTrak GUI panel by a green horizontal bar.	short
RangeDownLimit	Only applicable if Auto Ranging is selected in the RangeMode parameter above. This parameter sets the lower range limit as a percentage of the present range, 0 to 1000 = 0 to 100%. Similarly to RangeUpLimit, when the relative signal on a particular range drifts below the limit set in this parameter, the NanoTrak unit decrements the range to the next lower setting. The relative signal is displayed on the NanoTrak GUI panel by a green horizontal bar.	short
SettleSamples	Only applicable if Auto Ranging is selected in the RangeMode parameter above.	short

	This parameter determines the amount of averaging applied to the signal before autoranging takes place. Higher SettleSamples values improve the signal to noise ratio when dealing with noisy feedback signals. However, higher SettleSamples values also slow down the autoranging response. In a particular application, the SettleSamples value should be adjusted to obtain the best autoranging response combined with a noise free signal. Values are set in real world units, from '2' to '32', with a default setting value of '4'.	
RangeChangeType	Only applicable if Auto Ranging is selected in the RangeMode parameter above. This parameter specifies how range changes are implemented by the system. `0x01` `AUTORANGE_ALL` the unit visits all ranges when ranging between two input signal levels. `0x02` `AUTORANGE_ODD` only the odd numbered ranges between the two input signals levels will be visited. `0x03` `AUTORANGE_EVEN` only the even numbered ranges between the two input signals levels will be visited. These latter two modes are useful when large rapid input signal fluctuations are anticipated, because the number of ranges visited is halved to give a more rapid response.	word
RangeSW	Only applicable if Manual (SW) Ranging is selected in the RangeMode parameter above. The NanoTrak unit is equipped with an internal trans-impedance amplifier (TIA) circuit (and associated range/power level displays and control buttons in the GUI). This amplifier operates when an external input signal is connected to the Optical/PIN connector on the rear panel. There are 14 range settings (1 - 14) that can be used to select the best range to measure the input signal (displayed on the GUI panel relative input signal bar and display). **Note**. Range 1 and 2 (3 nA and 10 nA) are not applicable to TNA001 T-Cube units. This parameter returns the input signal range currently selected, defined as follows: **Range** **BNT, TNA, MNA** **KNA** **Returned** Range 1 3 nA 5 nA 0x03 Range 2 10 nA 16.6 nA 0x04 Range 3 30 nA 50 nA 0x05 Range 4 100 nA 166 nA 0x06 Range 5 300 nA 500 nA 0x07 Range 6 1 µA 1.65 µA 0x08 Range 7 3 µA 5.0 µA 0x09 Range 8 10 µA 16 µA 0x0A Range 9 30 µA 50 µA 0x0B Range 10 100 µA 166 µA 0x0C Range 11 300 µA 500 µA 0x0D Range 12 1 mA 1.66 m 0x0E Range 13 3 mA 5 mA 0x0F Range 14 10 mA N/A 0x10	word

Example

TX 30, 06, 0C, 00, D0, 01, 01, 00, 52, 03, 96, 00, 04, 00, 01, 00, 05, 00

Header: 30, 06, 0C, 00, D0, 01: Set_NTTIARangeParams, 12 byte data packet, Generic USB Device.

wRangeMode;	0x0001	Auto Ranging mode
sRangeUpLimit;	0x0352	850 == 85%
sRangeDownLimit;	0x0096	150 == 15%
wSettleSamples;	0x0004	4
wRangeChangeType;	0x0001	Auto range through all ranges
wRangeSW;	0x0005	P_PZ_NTTIA_RANGE30NANO

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
31	06	01	00	d	s

GET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
32	06	0C	00	d|	s	RangeMode		RangeUpLimit	

10	11	12	13	14	15	16	17
RangeDownLimit		SettleSamples		RangeChangeType		RangeSW	

See SET for structure

MGMSG_PZ_SET_NTGAINPARAMS 0x0633
MGMSG_PZ_REQ_NTGAINPARAMS 0x0634
MGMSG_PZ_GET_NTGAINPARAMS 0x0635

Function: This message sets the gain level of the NanoTrak control loop, and is used to ensure that the DC level of the input (feedback loop) signal lies within the dynamic range of the input. Increasing this value can lead to a more responsive NanoTrak behaviour as the signal variation around the circular path is enhanced. However, for a particular set up, if this value is too high, then unstable NanoTrak operation (indicated by a fluctuating circle) can result.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
33	06	04	00	d|	s	GainCtrlMode		NTGainSW	

Data Structure:

field	description	format
GainCtrlMode	This parameter is currently locked and cannot be changed: 0x02 GAIN_SW software setting gain control mode	word
NTGainSW	This parameter sets the loop gain, as a function of TIA range setting. The value is set between 100 and 10000 with a default value of 600. It is not normally necessary for anything other than minor adjustment from this default value.	short

Example: Set the NanoTrak loop gain to 600.

TX 33, 06, 04, 00, D0, 01, 02, 00, 58, 02

Header: 33, 06, 04, 00, D0, 01: Set_NTGainParams, 04 byte data packet, Generic USB Device.
GainCtrlMode 0x0002: Software Setting
NTGainSW 0x0258: 600

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
34	06	00	00	d	s

GET:
Command structure (10 bytes):

0	1	2	3	4	5	6	7	8	9
header						*Data*			
35	06	04	00	d|	s	GainCtrlMode		NTGainSW	

See SET for structure.

MGMSG_PZ_SET_NTTIALPFILTERPARAMS 0x0636
MGMSG_PZ_REQ_NTTIALPFILTERPARAMS 0x0637
MGMSG_PZ_GET_NTTIALPFILTERPARAMS 0x0638

Note – Not applicable to KNA101 units

Function: This message specifies the cut off frequency of the digital low pass (LP) filter applied to output readings of the internal amplifier (TIA) circuitry. If the readings displayed or returned are unstable, this setting can be used to remove any unwanted high frequency components and improve input signal stability.

SET:
Command structure (26 bytes)
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
36	06	14	00	d|	s	Param1				Param2			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Param3				Param4				Param5			

Data Structure:

field	description	format
FilterParams	This parameter contains low pass filter values which can be applied to the OUTPUT from the TIA, i.e. is applied to those reading params sent to the PC. It does NOT operate on the input to the TIA and does not operate on reading values used by the NanoTrak algorythms (these use a bandpass filter, effectively negating the need for a LP filter). The filter can be used to smooth out readings displayed in the GUI. It can also be used by client applications without affecting operation of the NanoTrak. **Note**. Although there are 5 parameters available, only the first parameter is used at this time. The filter can be set to OFF, or one of 5 frequency values as follows: Note. Only the first parameter is used at this time. 0 LP_NONE Low pass filter inactive 1 LP_1HZ Cut off all signals above 1Hz 2 LP_3HZ Cut off all signals above 3Hz 3 LP_10HZ Cut off all signals above 10Hz 4 LP_30HZ Cut off all signals above 30Hz 5 LP_100HZ Cut off all signals above 100Hz	long

Example: Set the LP filter to 1 Hz.

TX 36, 06, 14, 00, D0, 01, 05, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: 36, 06, 14, 00, D0, 01: Set_NTTIALPFilterParams, 20 byte data packet, Generic USB Device.
FilterParams: 05 LP_100HZ Cut off all signals above 100Hz

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
37	06	00	00	d	s

GET:
Command structure (26 bytes)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
38	06	14	00	d|	s	Param1				Param2			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Param3				Param4				Param5			

See SET for structure.

MGMSG_PZ_REQ_NTTIAREADING 0x0639
MGMSG_PZ_GET_NTTIAREADING 0x063A

Function: This message obtains the absolute signal value at the current position, in units as displayed on the GUI panel.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
39	06	00	00	d	s

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
3A	06	0A	00	d|	s	AbsReading				RelReading	

12	13	14	15
Data			
Range		UnderOverRead	

Data Structure:

field	description	format
AbsReading	This parameter returns the absolute TIA (PIN) current or BNC voltage value at the current position. The value is returned as a 4 byte floating point value in the range 1×10^{-9} to 1×10^{-3} (i.e. 1 nA to 1 mA or 1 to 10 V). The input source, TIA or BNC is set in the Set_NTFeedbackSRC message.	float
RelReading	The relative signal strength at the current position, in the range 0 to 32767 (i.e. 0 to 100% of the range currently selected). This value matches the length of the input signal bargraph on the GUI panel. (e.g. if the 3 µA range is currently selected, then a RelReading value of 16384 (50%) equates to 1.5 µA).).	word
Range	This parameter returns the input signal range currently selected. There are 14 range settings (1 - 14) that can be used to select the best range to measure the input signal (displayed on the GUI panel relative input signal bar and display). **Note**. Range 1 and 2 (3 nA and 10 nA) are not applicable to TNA001 T-Cube units. This parameter returns the input signal range currently selected, defined as follows: **Range** **BNT, TNA, MNA** **KNA** **Returned** Range 1 3 nA 5 nA 0x03 Range 2 10 nA 16.6 nA 0x04 Range 3 30 nA 50 nA 0x05	word

	Range 4 100 nA 166 nA 0x06 Range 5 300 nA 500 nA 0x07 Range 6 1 µA 1.65 µA 0x08 Range 7 3 µA 5.0 µA 0x09 Range 8 10 µA 16 µA 0x0A Range 9 30 µA 50 µA 0x0B Range 10 100 µA 166 µA 0x0C Range 11 300 µA 500 µA 0x0D Range 12 1 mA 1.66 m 0x0E Range 13 3 mA 5 mA 0x0F Range 14 10 mA N/A 0x10	
UnderOverRead	This parameter returns a value that identifies whether the unit is under reading or over reading the input signal as follows: 0x01 power signal is within current TIA range 0x02 power signal is under-reading for current TIA 0x03 power signal is over-reading for current TIA range e.g. if a user specified range of 3 µA is currently applied, this parameter returns '0x03' (Over read)' for input signals greater than 3 µA.	word

Example: Get the NanoTrak reading.

RX 3A, 06, 0A, 00, D0, 01, 00, 00, 00, 00, 00, 00, 05, 00, 01, 00

Header: 3A, 06, 0A, 00, D0, 01: Get_NTTIAReading, 10 byte data packet, Generic USB Device.

AbsReading	00, 00, 00, 00:		i.e. 20 nA
RelReading	00, 40:	16384,	i.e. 50%
Range	05, 00	Range 3,	i.e. 30 nA
UnderOverRead	01, 00	Within Range	

MGMSG_PZ_SET_NTFEEDBACKSRC 0x063B
MGMSG_PZ_REQ_NTFEEDBACKSRC 0x063C
MGMSG_PZ_GET_NTFEEDBACKSRC 0x063D

Function: This message sets the input source of the NanoTrak.
The INPUT_BNC settings are used when NanoTraking to optimise a voltage feedback signal. Typically, these inputs are selected when an external power meter which generates a voltage output, is connected to the rear panel SIG IN connector.

Note. In this case the internal amplifier circuit is bypassed and the 'Range' bar on the GUI panel is switched off (autoranging functionality is not required). Furthermore, although tracking occurs as normal, the tracking indicator on the GUI panel is inoperative.

The INPUT_TIA setting is used when NanoTraking to optimise a PIN current feedback signal. The TIA (trans impedence amplifier) input source should be selected when using the rear panel OPTICAL/PIN I/P connector with either an integral detector, or an external detector head connected to the optional SMB adapter. This option uses the internal amplifier circuit and associated functionality (e.g. autoranging).

SET:
Command structure (6 bytes)

0	1	2	3	4	5
header					
3B	06	00	00	d	s

The input source is set in byte 2 as follows:

P_PZ_NTFBTIA	0x01	TIA input
P_PZ_NTFBBNC1V	0x02	EXT input (1V range) (N/A for KNA101)
P_PZ_NTFBBNC2V	0x03	EXT input (2V range) (N/A for KNA101)
P_PZ_NTFBBNC5V	0x04	EXT input (5V range)
P_PZ_NTFBBNC10V	0x05	EXT input (10V range) (N/A for KNA101)

Example: Set the input source to TIA input.

TX, 3B, 06, 01, 00, 50, 01,

REQ:
Command structure (6 bytes)

0	1	2	3	4	5
header					
3C	06	00	00	d	s

GET:
Command structure (6 bytes)

0	1	2	3	4	5
header					
3D	06	00	00	d	s

See SET command for structure

MGMSG_PZ_REQ_NTSTATUSBITS 0x063E
MGMSG_PZ_GET_NTSTATUSBITS 0x063F

Function: Returns a number of status flags pertaining to the operation of the NanoTrak controller channel specified in the Chan Ident parameter. These flags are returned in a single 32 bit integer parameter and can provide additional useful status information for client application development. The individual bits (flags) of the 32 bit integer value are described in the following tables.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
3E	06	Chan Ident	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
3F	06	0A	00	d|	s	StatusBits					

Data Structure:

field	description	format
StatusBits	The status bits for the associated controller channel. The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TNA001 controller

Hex Value	Bit Number	Description
0x00000001	1	Tracking (1 - tracking, 0 - latched).
0x00000002	2	Tracking with Signal (1 – with signal, 0 – no signal)
0x00000004	3	Tracking Channel A (1 – Chan A only, 0 – Both channels)
0x00000008	4	T racking Channel B (1 – Chan B only, 0 – Both channels)
0x00000010	5	Auto-ranging (1 – auto ranging, 0 manual ranging).
0x00000020	6	Under Read (1 – under reading, 0 – reading within range).
0x00000040	7	Over Read (1 – over reading, 0 – reading within range).
	8 to 16	For future use
0x00010000	17	Channel A Connected (1 – Connected, 0 – Not Connected)
0x00020000	18	Channel B Connected (1 – Connected, 0 – Not Connected)
0x00040000	19	Channel A Enabled (1 – Enabled, 0 – Disabled)
0x00080000	20	Channel B Enabled (1 – Enabled, 0 – Disabled)
0x00100000	21	Channel A Control Mode (1 – Closed Loop, 0 – Open Loop)
0x00200000	22	Channel B Control Mode (1 – Closed Loop, 0 – Open Loop)
	23 to 32	For future use

BNT series controllers

Hex Value	Bit Number	Description
0x00000001	1	Tracking (1 - tracking, 0 - latched).
0x00000002	2	Tracking with Signal (1 – with signal, 0 – no signal)
0x00000004	3	Tracking Channel A (1 – Chan A only, 0 – Both channels)
0x00000008	4	T racking Channel B (1 – Chan B only, 0 – Both channels)
0x00000010	5	Auto-ranging (1 – auto ranging, 0 manual ranging).
0x00000020	6	Under Read (1 – under reading, 0 – reading within range).
0x00000040	7	Over Read (1 – over reading, 0 – reading within range).
	8 to 16	For future use
0x00010000	17	Channel A Connected (1 – Connected, 0 – Not Connected)
0x00020000		
0x00040000	19	Channel A Enabled (1 – Enabled, 0 – Disabled)
0x00080000	20	Channel B Enabled (1 – Enabled, 0 – Disabled)
0x00100000	21	Channel A Control Mode (1 – Closed Loop, 0 – Open Loop)
0x00200000	22	Channel B Control Mode (1 – Closed Loop, 0 – Open Loop)
Note. Bits 23 to 32 (Digital Input States) are only applicable if the associated digital input is fitted to your controller – see the relevant handbook for more details		
0x00100000	21	Digital input 1 state (1 - logic high, 0 - logic low).
0x00200000	22	Digital input 2 state (1 - logic high, 0 - logic low).
0x00400000	23	Digital input 3 state (1 - logic high, 0 - logic low).
0x00800000	24	Digital input 4 state (1 - logic high, 0 - logic low).
0x01000000	25	Digital input 5 state (1 - logic high, 0 - logic low).
0x02000000	26	Digital input 6 state (1 - logic high, 0 - logic low).
0x04000000	27	Digital input 7 state (1 - logic high, 0 - logic low).
0x08000000	28	Digital input 8 state (1 - logic high, 0 - logic low).
	29	For Future Use
0x20000000	30	Active (1 – indicates unit is active, 0 – not active)
0x40000000	31	For Future Use
0x80000000	32	Channel enabled (1 – enabled, 0- disabled)

MGMSG_PZ_REQ_NTSTATUSUPDATE 0x0664
MGMSG_PZ_GET_NTSTATUSUPDATE 0x0665

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the position and status of the controller (for example position and O/P voltage). The response will be sent by the controller each time the function is requested.

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
64	06	Chan Ident	00	d	s

GET:

Status update messages are received with the following format:-

Response structure (32 bytes)

6 byte header followed by 26 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
65	06	1A	00	d|	s	CircPosA		CircPosB		CircDia	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
AbsReading				RelReading		Range		UnderOverRead		StatusBits	

24	25	26	27	28	29	30	31
Data							
StatusBits		NTGain		PhaseCompA		PhaseCompB	

Data Structure:

field	description	format
CircPosA	The horizontal co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word
CircPosB	The vertical co-ordinate of the circle home position, in the range 0 to 65535 (0 to 100% of output voltage or 0 to 10 NanoTrak units).	word
CircDia	This NanoTrak scanning circle diameter. The diameter is returned in the range 0 to 65535, which relates to 0% to 100% output voltage –(i.e. 0 to 10 NT units).	word
AbsReading	The absolute TIA (PIN) current or BNC voltage value at the current position. The value is returned as a 4 byte floating point value in the range 1×10^{-9} to 1×10^{-3} (i.e. 1 nA to 1 mA or 1 to 10 V). The input source, TIA or BNC is set in the Set_NTFeedbackSRC message.	float

<table>
<tr><td>RelReading</td><td>The relative signal strength at the current position, in the range 0 to 32767 (i.e. 0 to 100% of the range currently selected). This value matches the length of the input signal bargraph on the GUI panel. (e.g. if the 3 µA range is currently selected, then a RelReading value of 16384 (50%) equates to 1.5 µA).</td><td>word</td></tr>
<tr><td>Range</td><td>The NanoTrak unit is equipped with an internal trans-impedance amplifier (TIS) circuit (and associated range/power level displays and control buttons in the GUI). This amplifier operates when an external input signal is connected to the Optical/PIN connector on the rear panel. There are 14 range settings (1 - 14) that can be used to select the best range to measure the input signal (displayed on the GUI panel relative input signal bar and display).
Note. Range 1 and 2 (3 nA and 10 nA) are not applicable to TNA001 T-Cube units.
This parameter returns the input signal range currently selected, defined as follows:
<table>
<tr><th>Range</th><th>BNT, TNA, MNA</th><th>KNA</th><th>Returned</th></tr>
<tr><td>Range 1</td><td>3 nA</td><td>5 nA</td><td>0x03</td></tr>
<tr><td>Range 2</td><td>10 nA</td><td>16.6 nA</td><td>0x04</td></tr>
<tr><td>Range 3</td><td>30 nA</td><td>50 nA</td><td>0x05</td></tr>
<tr><td>Range 4</td><td>100 nA</td><td>166 nA</td><td>0x06</td></tr>
<tr><td>Range 5</td><td>300 nA</td><td>500 nA</td><td>0x07</td></tr>
<tr><td>Range 6</td><td>1 µA</td><td>1.65 µA</td><td>0x08</td></tr>
<tr><td>Range 7</td><td>3 µA</td><td>5.0 µA</td><td>0x09</td></tr>
<tr><td>Range 8</td><td>10 µA</td><td>16 µA</td><td>0x0A</td></tr>
<tr><td>Range 9</td><td>30 µA</td><td>50 µA</td><td>0x0B</td></tr>
<tr><td>Range 10</td><td>100 µA</td><td>166 µA</td><td>0x0C</td></tr>
<tr><td>Range 11</td><td>300 µA</td><td>500 µA</td><td>0x0D</td></tr>
<tr><td>Range 12</td><td>1 mA</td><td>1.66 m</td><td>0x0E</td></tr>
<tr><td>Range 13</td><td>3 mA</td><td>5 mA</td><td>0x0F</td></tr>
<tr><td>Range 14</td><td>10 mA</td><td>N/A</td><td>0x10</td></tr>
</table></td><td>word</td></tr>
<tr><td>UnderOverRead</td><td>This parameter returns a value that identifies whether the unit is under reading or over reading the input signal as follows:
0x01 power signal is within current TIA range
0x02 power signal is under-reading for current TIA
0x03 power signal is over-reading for current TIA range
e.g. if a user specified range of 3 µA is currently applied, this parameter returns '0x03’ (Over read)' for input signals greater than 3 µA.</td><td>word</td></tr>
<tr><td>StatusBits</td><td>The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.</td><td>dword</td></tr>
<tr><td>NTGain</td><td>This parameter returns the loop gain, as a function of TIA range setting. The value is returned between 100 and 10000 (default value of 600).</td><td>short</td></tr>
<tr><td>PhaseCompA</td><td>The horizontal axis phase compensation value, returned in real world units as follows:-
value = (phase angle [degrees] / 360) * CircOscFreq
See the PZ_SET_NTCIRCPARAMS message for details on the CircOscFreq parameter</td><td>short</td></tr>
</table>

	Note. Negative phase values must be made positive by subtraction from 360 before the calculation is made.	
PhaseCompB	The vertical axis phase compensation value, returned in real world units as follows:- value = (phase angle [degrees] / 360) * CircOscFreq See the PZ_SET_NTCIRCPARAMS message for details on the CircOscFreq parameter **Note**. Negative phase values must be made positive by subtraction from 360 before the calculation is made.	short

TNA001 controller

Hex Value	Bit Number	Description
0x00000001	1	Tracking (1 - tracking, 0 - latched).
0x00000002	2	Tracking with Signal (1 – with signal, 0 – no signal)
0x00000004	3	Tracking Channel A (1 – Chan A only, 0 – Both channels)
0x00000008	4	T racking Channel B (1 – Chan B only, 0 – Both channels)
0x00000010	5	Auto-ranging (1 – auto ranging, 0 manual ranging).
0x00000020	6	Under Read (1 – under reading, 0 – reading within range).
0x00000040	7	Over Read (1 – over reading, 0 – reading within range).
	8 to 16	For future use
0x00010000	17	Channel A Connected (1 – Connected, 0 – Not Connected)
0x00020000	18	Channel B Connected (1 – Connected, 0 – Not Connected)
0x00040000	19	Channel A Enabled (1 – Enabled, 0 – Disabled)
0x00080000	20	Channel B Enabled (1 – Enabled, 0 – Disabled)
0x00100000	21	Channel A Control Mode (1 – Closed Loop, 0 – Open Loop)
0x00200000	22	Channel B Control Mode (1 – Closed Loop, 0 – Open Loop)
	23 to 32	For future use

BPC series controllers

Hex Value	Bit Number	Description
0x00000001	1	Piezo actuator connected (1 - connected, 0 - not connected).
	2 to 4	For Future Use
0x00000010	5	Piezo channel has been zero'd (1 - zero'd, 0 not zero'd).
0x00000020	6	Piezo channel is zeroing (1 - zeroing, 0 - not zeroing).
0x00000040	7 to 8	For Future Use
0x00000100	9	Strain gauge feedback connected (1 - connected, 0 - not connected).
	10	For Future Use
0x00000400	11	Position control mode (1 - closed loop, 0 - open loop).
	12 to 20	For Future Use
Note. Bits 21 to 28 (Digital Input States) are only applicable if the associated digital input is fitted to your controller – see the relevant handbook for more details		
0x00100000	21	Digital input 1 state (1 - logic high, 0 - logic low).
0x00200000	22	Digital input 2 state (1 - logic high, 0 - logic low).
0x00400000	23	Digital input 3 state (1 - logic high, 0 - logic low).
0x00800000	24	Digital input 4 state (1 - logic high, 0 - logic low).
0x01000000	25	Digital input 5 state (1 - logic high, 0 - logic low).
0x02000000	26	Digital input 6 state (1 - logic high, 0 - logic low).
0x04000000	27	Digital input 7 state (1 - logic high, 0 - logic low).

0x08000000	28	Digital input 8 state (1 - logic high, 0 - logic low).
	29	For Future Use
0x20000000	30	Active (1 – indicates unit is active, 0 – not active)
0x40000000	31	For Future Use
0x80000000	32	Channel enabled (1 – enabled, 0- disabled)

MGMSG_PZ_ACK_NTSTATUSUPDATE 0x0666

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called "server alive" must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
66	06	00	00	d	s

TX 66, 06, 00, 00, 50, 01

MGMSG_KNA_SET_NTTIALPFILTERCOEFFS 0x0687
MGMSG_KNA_REQ_NTTIALPFILTERCOEFFS 0x0688
MGMSG_KNA_GET_NTTIALPFILTERCOEFFS 0x0689

Function: This message specifies the cut off frequency of the digital low pass (LP) filter applied to output readings of the internal amplifier (TIA) circuitry. If the readings displayed or returned are unstable, this setting can be used to remove any unwanted high frequency components and improve input signal stability.

SET:

Command structure (26 bytes)

6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
87	06	14	00	d|	s	Param1				Param2			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Param3				Param4				Param5			

Data Structure:

field	description	format
FilterParams	This parameter contains low pass filter values which can be applied to the OUTPUT from the TIA, i.e. is applied to those reading params sent to the PC. It does NOT operate on the input to the TIA and does not operate on reading values used by the NanoTrak algorithms (these use a bandpass filter, effectively negating the need for a LP filter). The filter can be used to smooth out readings displayed in the GUI. It can also be used by client applications without affecting operation of the NanoTrak. **Note**. Although there are 5 parameters available, only the first parameter is used at this time. The filter can be set to OFF, or one of 5 frequency values as follows: Note. Only the first parameter is used at this time. 0 LP_NONE Low pass filter inactive 1 LP_1HZ Cut off all signals above 1Hz 2 LP_3HZ Cut off all signals above 3Hz 3 LP_10HZ Cut off all signals above 10Hz 4 LP_30HZ Cut off all signals above 30Hz 5 LP_100HZ Cut off all signals above 100Hz	long

Example: Set the LP filter to 1 Hz.

TX 87, 06, 14, 00, D0, 01, 05, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: 87, 06, 14, 00, D0, 01: Set_NTTIALPFilterParams, 20 byte data packet, Generic USB Device.
FilterParams: 05 LP_100HZ Cut off all signals above 100Hz

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
88	06	00	00	d	s

GET:
Command structure (26 bytes)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
89	06	14	00	d|	s	Param1				Param2			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
Param3				Param4				Param5			

See SET for structure.

MGMSG_KNA_SET_KCUBEMMIPARAMS 0x068A
MGMSG_KNA_REQ_KCUBEMMIPARAMS 0x068B
MGMSG_KNA_GET_KCUBEMMIPARAMS 0x068C

Function: Used to set the intensity of the LCD display on the TOP of the KNA101 unit. Intensity is set as a percentage of full brightness in the range 0 (off) to 100%. Also used to set the display time out and dim level as described below.

SET:

Command structure (22 bytes)

6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
8A	06	10	00	d|	s	WheelStep		DispBrightness		Reserved		Reserved	

14	15	16	17	18	19	20	21
Data							
Reserved		Reserved		Reserved		Reserved	

Data Structure:

field	description	format
WheelStep	Sets the adjustment rate of the top panel wheel as follows: 0 – Low 1 – Mid 2 - High	word
DispBrightness	In certain applications, it may be necessary to adjust the brightness of the LCD display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again.	word

Example: Set the Wheel Adjustment rate to High, and the Display intensity 50%.

TX 8A, 06, 10, 00, D0, 01, 02, 00, 32, 00,

Header: F6, 07, 04, 00, D0, 01: Set_KCUBEMMIPARAMS, 16 byte data packet, Generic USB Device.

WheelStep: 02, 00: Sets the wheel adjustment rate to High

DispIntensity: 32, 00: Sets the display brightness to 50%

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
8B	06	01	00	d	s

Example: Request the display intensity

TX 8B, 06, 01, 00, 50, 01

GET:

Command structure (22 bytes)

6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
8C	06	10	00	d|	s	WheelStep		DispBrightness		Reserved		Reserved	

14	15	16	17	18	19	20	21
Data							
Reserved		Reserved		Reserved		Reserved	

See SET for data structure.

MGMSG_KNA_SET_KCUBETRIGIOCONFIG 0x068D
MGMSG_KNA_REQ_KCUBETRIGIOCONFIG 0x068E
MGMSG_KNA_GET_KCUBETRIGIOCONFIG 0x068F

Function: The KNA101 K-Cube NanoTrak has two bidirectional ports (IO1 and IO2). Both ports can be configured as a trigger input to respond to an external signal, or as a trigger output to control an external circuit. Additionally, IO1 can be used as an external input while IO2 is used as an external output.
When the port is used as a trigger output it provides a push-pull drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuity. The active logic state can be selected High or Low to suit the requirements of the application.

This message sets the operating parameters of the IO1 and IO2 connectors on the front panel of the unit.

Warning. Do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

Trigger Modes
Input Trigger Modes
When configured as an input, the TRIG ports can be used as a general purpose digital input, or for starting a track or home event as follows:

0x00 The trigger IO is disabled.
0x01 General purpose logic input (read through status bits using the PZ_GET_NTSTATUSUPDATE message).
0x02 Input trigger for Tracking. On receipt of the trigger, the unit starts to track the max coupled power signal.
0x03 Input trigger for Home. On receipt of the trigger, the unit drives the circle to the home position, as set using the Set_NTCircHomePos message.

When used for triggering, the port is edge sensitive. In other words, it has to see a transition from the inactive to the active logic state (Low->High or High->Low) for the trigger input to be recognized. For the same reason a sustained logic level will not trigger repeated events. The trigger input has to return to its inactive state first in order to start the next trigger.

Output Trigger Modes
When configured as an output, the TRIG ports can be used as a general purpose digital output, or for triggering an external circuit when tracking is active.

0x0A General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).
0x0B Tracking Active. When tracking is active, the unit outputs a 5V signal for use in external circuits, e.g. a warning light.

Trigger Polarity

The polarity of the trigger pulse is specified in the TPolarity parameters as follows:

0x01 The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).

0x02 The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

SET:

Command structure (26 bytes)

6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
8D	06	14	00	d|	s	T1Mode		T1Polarity		T1Par	

12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data													
T2Mode		T2Polarity		T2Par		Reserved		Reserved		Reserved		Reserved	

Data Structure:

field	description	format
T1Mode	TRIG1 operating mode:	word
T1Polarity	The active state of TRIG1 (i.e. logic high or logic low) .	word
T1Par	Not Used	word
T2Mode	TRIG2 operating mode:	word
T2Polarity	The active state of TRIG2 (i.e. logic high or logic low) .	word
T2Par	Not Used	word

Example: Set the Trigger parameters for KNA101 as follows:
T1Mode – TrigIn – Start Tracking
T1Polarity – High
T2Mode – Disabled
T2Polarity – N/A

TX 8D, 06, 14, 00, D0, 01, 02, 00, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: 8D, 06, 14, 00, D0, 01: Set_KCube_TrigIOConfig, 20 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).

T1Mode – 02, 00 TrigIn_Start Tracking
T1Polarity – 01,00 High
T2Mode – 00,00 Disabled
T2Polarity – 00,00 N/A

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
8E	06	01	00	d	s

GET:
Command structure (26 bytes)
6 byte header followed by 20 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
8D	06	14	00	d|	s	T1Mode		T1Polarity		T1Par	

12	13	14	15	16	17	18	19	20	21	22	23	24	25
Data													
T2Mode		T2Polarity		T2Par		Reserved		Reserved		Reserved		Reserved	

See SET message for structure.

MGMSG_KNA_REQ_XYSCAN 0x06A0
MGMSG_KNA_GET_XYSCAN 0x06A1
MGMSG_KNA_STOP_XYSCAN 0x06A2

Note. These messages are applicable only to KNA101 units, and can be used only when operating in Piezo Mode – see MGMSG_PZ_SET_NTMODE.

Function: In some applications, it may be useful to know roughly where the high power region is located within the range of the piezo device (e.g. to avoid power optimization on a side peak). When this message is called, the K-Cube unit moves the stage in an XY raster scan pattern over the full piezo range, and measures the optical power in a grid 96 x 96 points. The power data is then returned as a measure of intensity at each point, in the range 0 to 255. During the scan, auto-ranging is disabled and the range is locked at the range setting in use when the scan was requested. The data is also shown on the LCD display or GUI panel as a power intensity map, 96 x 96 pixels.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
A0	06	01	00	d	s

Example: Request the XY Scan

TX 90, 06, 01, 00, 50, 01

GET:
Command structure (106 bytes)
6 byte header followed by 100 byte data packet as follows:

0	1	2	3	4	5	6	7	......	....	....	....	104	105
header						*Data*							
A1	06	64	00	d|	s	Line Number		Range		96 byte intensity map			

Data Structure

field	description	format
Line Number	When the message is called it runs 96 times, once for each line on the Y axis. Each run captures 96 data points on the X axis. This parameter specifies the Y axis line in the raster scan, in the range 0 to 95.	word
Range	The NanoTrak unit is equipped with an internal trans-impedance amplifier (TIA) circuit (and associated range/power level displays and control buttons in the GUI). This amplifier operates when an external input signal is connected to the Optical/PIN connector	word

	on the rear panel. The KNA unit has 13 range settings that can be used to select the best range to measure the input signal (displayed on the GUI panel relative input signal bar and display). During the scan, auto-ranging is disabled and the range is locked and this parameter returns the range setting in use when the scan was requested. **Range** **Limit** **Returned** Range 1 5 nA 0x03 Range 2 16.6 nA 0x04 Range 3 50 nA 0x05 Range 4 166 nA 0x06 Range 5 500 nA 0x07 Range 6 1.65 µA 0x08 Range 7 5.0 µA 0x09 Range 8 16 µA 0x0A Range 9 50 µA 0x0B Range 10 166 µA 0x0C Range 11 500 µA 0x0D Range 12 1.66 mA 0x0E Range 13 5 mA 0x0F	
Intensity Map	96 bytes. Each byte represents the intensity at a given point on the X-axis, in the range 0 to 255.	

MGMSG_NT_SET_EEPROMPARAMS 0x07E7

Function: Used to save the parameter settings for the specified message. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the 'Settings' button found in the lower right hand corner of the user interface).

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
E7	07	04	00	d|	s	Chan Ident		MsgID	

Data Structure:

field	description	format
Chan Ident	The channel being addressed	word
MsgID	The message ID of the message containing the parameters to be saved.	word

Example:

TX E7, 07, 04, 00, D0, 01, 01, 00, 18, 06,

Header: E7, 07, 04, 00, D0, 01: Set_EEPROMPARAMS, 04 byte data packet, Generic USB Device.
Chan Ident: 01, 00: Channel 1
MsgID: Save parameters specified by message 0618 (SetNTCircParams).

MGMSG_NT_SET_TNA_DISPSETTINGS 0x07E8
MGMSG_NT_REQ_TNA_DISPSETTINGS 0x07E9
MGMSG_NT_GET_TNA_DISPSETTINGS 0x07EA

Function: Used to set the intensity of the LED display on the front of the TNA and KNA units.

SET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
E8	07	02	00	d|	s	DispIntensity	

Data Structure:

field	description	format
DispIntensity	The intensity is set as a value from 0 (Off) to 255 (brightest).	word

Example: Set the input source to software and potentiometer.

TX E8, 07, 02, 00, D0, 01, 64, 00,

Header: E8, 07, 02, 00, D0, 01: Set_DISPSETTINGS, 02 byte data packet, Generic USB Device.
DispIntensity: 64, 00: Sets the display brightness to 100 (40%)

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
E9	07	01	00	d	s

Example: Request the display intensity

TX E9, 07, 01, 00, 50, 01

GET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
EA	07	02	00	d|	s	DispIntensity	

See SET for data structure.

MGMSG_NT_SET_TNAIOSETTINGS 0x07EB
MGMSG_NT_REQ_TNAIOSETTINGS 0x07EC
MGMSG_NT_GET_TNAIOSETTINGS 0x07ED

Note. Applicable only to TNA T-Cube and KNA K-Cube Units.

Function: This message is used to set parameters which control the NanoTrak output signal ranges and the way in which these signals are routed to the associated external drivers.

SET:

Command structure (14 bytes)

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
EB	07	04	00	d|	s	LVOutRange		LVOutRoute		HVOutRange		SignlORoute	

Data Structure:

field	description	format
LVOutRange	**TNA001 Units:** The output signals from the NanoTrak T-Cube are routed to the piezo drivers to position the piezo actuators. Earlier piezo T-cubes accept a 5V input while later cubes accept a 10V input. Other piezo amplifiers with 5V or 10V input ranges may be driven from the NanoTrak T-Cube. This parameter sets the LV output range as follows: 0x01 0 to 5V Output Range 0x02 0 to 10V Output Range **KNA101 Units:** The internal piezo drivers of the KNA unit are limited to an output current of around 5 mA, which is insufficient for some of the higher circle scanning frequencies available. In this case it will be necessary to route the output signals from the NanoTrak K-Cube to an external piezo driver. This parameter fixes the LV output range at 10 V (parameter value 0x02) and cannot be adjusted.	word
LVOutRoute	**TNA001 Units:** This parameter sets the way the signals are routed to the piezo T-Cubes as follows: 0x01 Rear panel SMA connectors only 0x02 Rear panel SMA connectors and Hub routing **KNA101 Units:** This parameter is fixed to route signals via the front and rear panel external SMA connectors and cannot be adjusted. Signals cannot be routed to external piezo drivers via the hub.	word
HVOutRange	**KNA101 Units only:** The piezo actuator connected to the unit has a specific maximum operating voltage range. This parameter sets the maximum piezo drive voltage from the HV Out connectors. The LSB relates to Chan 1 and the next bit relates to Chan 2 as follows: Chan 1: 0 = 75V and 1 = 150V, Chan 2: 0 = 75V and 10 = 150V Example: To set both channels to 150V output – 0000 1001	word

SignIORoute	**KNA101 Units only:** The IO1 connector on the front panel can be configured as an external input and IO2 as an external output. This parameter specifies the function of these connectors. The LSB relates to Chan 1 and the next bit relates to Chan 2 as follows: **IO1** 0 – IO 1 is disabled and the power signal is input via the PIN OPTICAL INPUT connector on the rear panel 1 – IO 1 is enabled, and the power signal is input via this SMA connector. **IO2** 0 – IO 2 is disabled 10 – IO 2 is enabled and the power signal is output as a 0 to 10V signal via this SMA connector Example. Set IO 1 to disabled and IO2 to enabled – 00,00 10,00 **AC BOOST** At low signal levels, when scanning for optical power a small change in circle position can result in a large change in power reading. As the search gets closer to the max power position, changes in circle position result in only small changes in power reading. The AC BOOST function amplifies the difference in power reading to better emphasise the direction of max power. This function is activated by setting the 3rd bit of the parameter to 100 Example. Set IO 1 to disabled and IO2 to enabled and AC Boost active – 01,00 10,00	word

Example

Tx EB,07,08,00,D0,01, 02,00,01,00,01,10,00,10

Header: EB, 07, 08, 00, D0, 01: Set_TNAIOSettings, 08 byte data packet, Generic USB Device.
LVOutRange: 02, 00: 0 to 5V range
LVOutRoute: 01, 00: Signal routing via rear panel SMA connectors.
HVOutRange: 01, 10: Ch1 and CH2 to 150V
SignIORoute: 00, 10: IO1 disabled, IO 2 enabled.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
EC	07	Chan Ident	00	d	s

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
ED	07	04	00	d|	s	LVOutRange		LVOutRoute		HVOutRange		SignIORoute	

See SET for structure.

Laser Control Messages

Introduction

The ‘Laser’ ActiveX Control provides the functionality required for a client application to control one or more Laser devices.
The methods of the Laser Control Object can then be used to control the T-Cube Laser Source (TLS001) and Laser Driver (TLD001) units, and the K-Cube Laser source (KLS101).
Activities such as switching between display modes, setting the laser power set point, reading the laser power or current and setting the LED display intensity can be performed.
For details on the use of the Laser Source, refer to the handbook supplied with the unit.

MGMSG_LA_SET_PARAMS 0x0800
MGMSG_LA_REQ_PARAMS 0x0801
MGMSG_LA_GET_PARAMS 0x0802

Function: This generic parameter set/request message is used to control all the functionality of the TLD001, KLD101, TLS001, KLS635 and KLS1550. The specific parameters to control are identified by the use of sub-messages. These sub messages comply with the general format of the Thorlabs message protocol but rather than having a unique first and second byte in the header carrying the "message identifier" information, the first and second byte remain the same.
Instead, for the SET and GET messages, the message identifier is carried in the first two bytes in the data packet part of the message, whilst for the REQ message it is encoded as the third byte of the header.
Likewise, when the unit responds, the first two bytes of the response remain the same and the first two bytes of the data packet identify the sub-message to which the information returned in the remaining part of the data packet relates.
The following sub messages are applicable to all units:

Set/Request/Get Laser Power Setpoint (sub-message ID = 1)
Request/Get Laser Current and Power (sub-message ID = 3)
Set/Request/Get Laser Power Control Source (sub-message ID = 5)
Request/Get Status Bits (sub-message ID = 7)
Request/Get Maximum TLS001 Limits (sub-message ID = 9)
Request/Get Maximum TLD001 Laser Current (sub-message ID = 0A)
Set/Request/Get Display Settings (sub-message ID = 0B)
Set/Request/Get Misc TLD001 Settings (sub-message ID = 0D)
Set/Request/Get MMI Parameters (sub-message ID = 0E)
Set/Request/Get KLDDigOutputs (sub-message ID = 11)

to explain the principle, the following examples describe the first of these messages in more detail.

Example - Set/Request/Get Laser Power Setpoint (sub-message ID = 1)
This sub-message is not applicable to TLD001 Laser Driver units.

This sub-command is used to set / read the laser power setpoint. The setpoint is the required laser power that the TLS001 and KLS units will attempt to maintain. This is not necessarily the same as the actual laser power because if the current limit for the laser diode is exceeded, the setpoint will not be reached.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
00	08	04	00	d|	s	MsgID		SetPoint	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
SetPoint	The Laser power setpoint (0 to 32767 -> 0% to 100% power).to be saved.	word

Example: Set the laser power setpoint to be set to 5% of the maximum power

TX 00, 08, 04, 00, D0, 01, 01, 00, 66, 06,

Header: 00, 08, 04, 00, D0, 01: Set_PARAMS, 04 byte data packet, Generic USB Device.
MsgID: 01, 00: Set Laser Power Setpoint
SetPoint:.66, 06: the laser power setpoint, 0x0666 (1638 decimal), which is 5 % of the full power.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	01	00	d	s

TX 01, 08, 01, 00, 50, 01,

GET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
02	08	04	00	d|	s	MsgID		SetPoint	

See SET message for data structure

Example - Request/Get Laser Current and Power (sub-message ID = 3)
This sub-message is not applicable to TLD001 Laser Driver units.
This sub-command is used to read the actual laser power and the laser current. Note that there is no SET message as only the setpoint power can be set, not the actual power or current.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	03	00	d	s

TX 01, 08, 03, 00, 50, 01,

GET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
02	08	06	00	d|	s	MsgID		LaserCurrent		LaserPower	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
LaserCurrent	The Laser current (0 to 32767 -> 0 to max current in mA)	word
LaserPower	The Laser power (0 to 32767 -> 0% to 100% power)	word

Example: Get the laser current and power

RX 02, 08, 06, 00, D0, 01, 03, 00, 66, 06, 66, 06

Header: 00, 08, 06, 00, D0, 01: Set_PARAMS, 06 byte data packet, Generic USB Device.
MsgID: 03, 00: Get Laser Current and Power
LaserCurrent:.66, 06: the laser current, 0x0666 (1638 decimal), which is 5 mA for a 100 mA max current laser.
LaserPower:.66, 06: the laser power, 0x0666 (1638 decimal), which is 5% of the full power.

Example - Request/Get Laser Current and Power (sub-message ID = 4)
This sub-message is applicable only to TLD001 Laser Driver units.
This sub-command is used to read the actual laser power and the laser current. Note that there is no SET message as only the setpoint power can be set, not the actual power or current.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	04	00	d	s

TX 01, 08, 04, 00, 50, 01,

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
02	08	06	00	d|	s	MsgID		LaserCurrent		LaserPower		LaserVoltage	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
LaserCurrent	The Laser current (-32768 to 32767 -> -200 to 200 mA)	word
LaserPower	The Laser power (0 to 32767 -> 0% to TIA Range Max in mA)	word
LaserVoltage	The Laser forward voltage (-10000 to 10000 -> _10.0 V to 10.0 V)	word

Example: Get the laser current and power

RX 02, 08, 08, 00, D0, 01, 04, 00, 66, 06, 66, 06, 88, 13

Header: 02, 08, 08, 00, D0, 01: Set_PARAMS, 08 byte data packet, Generic USB Device.
MsgID: 04, 00: Get Laser Current and Power
LaserCurrent:.66, 06: the laser current, 0x0666 (1638 decimal), which is 5 mA for a 100 mA max current laser.
LaserPower:.66, 06: the laser power, 0x0666 (1638 decimal), which is 5% of the full power.
LaserVoltage:.88, 13: the laser voltage, 0x1388 (5000 decimal), which is 5V

Example - Set/Request/Get the Laser Power Control Source (sub-message ID = 5)
This sub-command is used to set / read the laser power control source. The laser power can be controlled by software commands, the potentiometer on the top of the unit or the external SMA input. Only one control source can be active at any time, the options are mutually exclusive.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
00	08	04	00	d|	s	MsgID		LaserSource	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
LaserSource	The Laser power source. This parameter is different depending on which unit is being address, as follows… **TLD** 1 = Software control only 2 = External source via SMA connector only 4 = Potentiometer only **TLS** 0 = Software control only 1 = External source via SMA connector only 4 = Potentiometer only **KLD and KLS** 0 = Software control only 1 = External source via SMA connector only 4 = Top panel wheel and Software 8 = Reserved	word

Example: Set the laser power source to be external SMA input on a TLS001 unit.

TX 00, 08, 04, 00, D0, 01, 05, 00, 01, 00

Header: 00, 08, 04, 00, D0, 01: Set_PARAMS, 04 byte data packet, Generic USB Device.
MsgID: 05, 00: Set Laser Power Source
LaserSource:.01, 00: the laser power source is the external SMA input.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	05	00	d	s

TX 01, 08, 01, 00, 50, 01,

GET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
02	08	04	00	d|	s	MsgID		LaserSource	

See SET message for data structure

Request/Get Status Bits (sub-message ID = 7)
This sub command can be used to request the status bits. The message only has a request/ get part.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	07	00	d	s

TX 01, 08, 07, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
02	08	06	00	d|	s	MsgID		StatusBits			

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
StatusBits	The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TLS001 controller

Hex Value	Bit Number	Description
0x00000001	1	Laser output enabled state (1 - enabled, 0 - disabled).
0x00000002	2	Keyswitch enabled state (1 - enabled, 0 – disabled)
0x00000004	3	Laser control mode (1 - power [closed loop], 0 - current [open loop])
0x00000008	4	Safety interlock, (1 - enabled, 0 – disabled)
0x00000010	5	Units mode (1 - mA, else 0).
0x00000020	6	Units mode (1 - mW, else 0).
0x00000040	7	Units mode (1 - dBm, else 0)
	8	For Future Use

Example

RX 02, 08, 06, 00, 81, 50, 07, 00, 2B, 00, 00, 00

Header: 02, 08, 06, 00, 81, 50: LA_Get_Params, 06 byte data packet, Generic USB Device.
MsgID: 07, 00: Get Status Bits
StatusBits: 2B,00,00,00, i.e. 00101011 the display shows mW units, the safety interlock is enabled, the keyswitch is enabled and the output is enabled.

Request/Get Maximum Limits (sub-message ID = 9)
This sub-message is not applicable to TLD001 Laser Driver units.
This sub command can be used to request the maximum limits of the laser source, such as maximum current, maximum power and the wavelength of the laser diode. The message only has a request/ get part.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	09	00	d	s

TX 01, 08, 09, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
02	08	08	00	d|	s	MsgID		MaxCurrent		MaxPower		Wavelength	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
MaxCurrent	The Laser max current (0 to 65535 -> 0 to655.35 mA)	word
MaxPower	The Laser max power (0 to 65535 -> 0 to 6.5535 mW)	word
WaveLength	The Laser wavelength in nm (635 or 1550)	word

Example – Get Laser Limits
RX 02, 08, 08, 00, D0, 01, 09, 00, C8, 00, 05, 00, 0E, 06

Header: 00, 08, 06, 00, D0, 01: Set_PARAMS, 06 byte data packet, Generic USB Device.
MsgID: 09, 00: Get Laser Max Limits
MaxCurrent:.C8, 00:, 0x00C8 i.e. 200mA max current.
MaxPower:.05, 00:, 0x0005 i.e. 5 mW max power.
Wavelength:.0E, 06: the laser power, 0x060E (1550 decimal), wavelength 1550 nm.

Request/Get Maximum Laser Diode Current (sub-message ID = 10 [0A])
This sub-message is applicable only to TLD001 Laser Diode Driver units.
This sub command can be used to request the TLD001 maximum laser diode current. The message only has a request/ get part.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	0A	00	d	s

TX 01, 08, 0A, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
02	08	04	00	d|	s	MsgID		MaxCurrent	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
MaxCurrent	The Laser max current (-32768 to 32767 -> -Min mA to Max mA)	word

Example – Get Laser Limits
RX 02, 08, 04, 00, D0, 01, 0A, 00, C8, 00, 05, 00, 0E, 06

Header: 02, 08, 04, 00, D0, 01: Set_PARAMS, 04 byte data packet, Generic USB Device.
MsgID: 0A, 00: Get Laser Max Limits
MaxCurrent:.C8, 00:, 0x00C8 i.e. 200mA max current.

Set/Request/Get Display Settings (sub-message ID = 11 [0B])

This message can be used to adjust or read the front panel LED display brightness and the display units. **It is not applicable to KLSxxx units.**

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
00	08	08	00	d|	s	MsgID		DispIntensity		DispUnits		Unused	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
DispIntensity	The intensity is set as a value from 0 (Off) to 255 (brightest).	word
DispUnits	The LED display window on the front of the unit can be set to display the laser output in mA, mW or dBm as follows. 1 display shows laser current in mA. 2 display shows laser power in mW. 3 display shows laser power in dBm (relative to 1 mW)	word
Unused	N/A	word

Example: Set the display to show the laser current in Amps and at max brightness:

TX 00, 08, 08, 00, D0, 01, 0B, 00, FF, 00, 01, 00, 00, 00

Header: 00, 08, 08, 00, D0, 01: Set_Params, 08 byte data packet, Generic USB Device.
MsgID: 0B, 00: Set Display Settings
DispIntensity: FF, 00: Sets the display brightness to 255 (100%)
DispUnits: 01, 00: Sets the display units to mA

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	0B	00	d	s

Example: TX 01, 08, 0B, 00, 50, 01

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
02	08	08	00	d|	s	MsgID		DispIntensity		DispUnits		Unused	

See SET for data structure.

Set/Request/Get Miscellaneous Laser Driver Parameters (sub-message ID = 13 [0D])
This message is applicable only to TLD001 Laser Diode Driver units.

Each laser diode has specific relationship between the output power and the photodiode current. This message sets the polarity and the calibration factor for converting between output power and the photodiode current.
The calibration factor for the type of laser diode being used is set in the WACalibFactor parameter. For example, if set to 10, a photodiode current of 1mA produces an output power of 10mW.
The calibration factor for the particular laser diode being used should be quoted in the associated data sheet. If this is not available, then a test calibration should be performed, using a power meter to measure the output for a known photodiode current.
Laser diodes are manufactured in a variety of packages and pin configurations, with or without an internal photodiode. In addition, normally one terminal of the laser diode is connected to the metal case and commoned with either the anode or cathode of the photodiode. This can be established from the laser diode data sheet and the device should be connected to the laser driver accordingly.

This message configures the unit for either an anode grounded or a cathode grounded diode. The polarity of the laser diode connected to the TLD001 unit is specified in the LaserPolarity parameter.
By default, when the output is enabled, the laser current will be increased immediately to max current. If required, the output current can be increased gradually in steps 10% of selected max current output. This option is set in the Rampup parameter.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
00	08	08	00	d|	s	MsgID		WACalibFactor			

12	13	14	15
Data			
LaserPolarity		Rampup	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters	word
WACalibFactor	The calibration factor used to convert photo diode current (IPD) to output laser power (PLD).	float
LaserPolarity	The laser diode connection polarity as follows. 1 cathode grounded 2 anode grounded	word
Rampup	The method of energizing the laser. 1 Rampup selected - the output current is increased gradually in steps 10% of selected max current output	word

Example: Set the unit to have a calibration factor of 10, for a cathode grounded laser diode:

TX 00, 08, 08, 00, D0, 01, 0D, 00, 0A, 00, 00, 00, 01, 00, 00, 00

Header: 00, 08, 08, 00, D0, 01: Set_Miscellaneous Params, 08 byte data packet, Generic USB Device.
MsgID: 0D, 00: Set Miscellaneous Parameters
WACalibFactor: 0A, 00, 00, 00: Sets the calibration factor to 10
LaserPolarity: 01, 00: Sets the polarity to Cathode Grounded
Rampup: 00, 00: The laser current is increased immediately to maximum.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	0B	00	d	s

Example: TX 01, 08, 0D, 00, 50, 01

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
02	08	08	00	d|	s	MsgID		WACalibFactor			

12	13	14	15
Data			
LaserPolarity		Unused	

See SET for data structure.

Set/Request/Get MMI Parameters (sub-message ID = 14 [0E])
Applicable only to KLSxxx units.
This message can be used to adjust or read the front panel LED display brightness.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
00	08	08	00	d|	s	SubMsgID		DispIntensity		For Future Use			

14	15
Data	

Data Structure:

field	description	format
MsgID	The message ID (i.e. 0E00) of the message containing the parameters	word
DispIntensity	The intensity is set as a percentage of maximum brightness, from 20 (dimmest) to 100 (brightest).	word

Example: Set the display to max brightness,
TX 00, 08, 08, 00, D0, 01, 0B, 00, 64, 00, 00, 00, 00, 00

Header: 00, 08, 0A, 00, D0, 01: LA_SetParams, 08 byte data packet, Generic USB Device.
SubMsgID: 0E, 00: Set Display Settings
DispIntensity: 64, 00: Sets the display brightness to 100%

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
01	08	0B	00	d	s

Example: TX 01, 08, 0E, 00, 50, 01

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
02	08	08	00	d|	s	SubMsgID		DispIntensity		For Future Use			

14	15
Data	

See SET for data structure.

Set/Request/Get LA_KLDDigOutputs (sub-message ID =17 (0x11)

This sub-message is applicable only to KLD101 units.

Used to set the digital outputs of the KLD101 unit, if the trigger port is to be used as a general purpose digital output (i.e. trigger mode set to 0x0A TRIGOUT_GPO).
The logic state of the output can be inverted by setting the Triggering Polarity parameter to "Low"; with this option selected the state of the output will be the opposite of the corresponding bit setting in the software call. The default state of the output in this mode is also the opposite of the option selected as the Triggering Polarity.

SET
Command structure (12bytes)
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
00	08	06	00	d	s	SubMsgID		DigOPs		Reserved	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 1100) of the message containing the parameters	word
DigOPs	The status of the digital outputs. The lowest two bits relate to TRIG1 and TRIG2	word
Reserved		

Example: Set both Digital Outputs to ON:

TX 00, 08, 06, 00, D0, 01, 10, 00, 11, 00, 00, 00,

Header: 00, 08, 06, 00, D0, 01: LA_SetParams, 6 byte data packet, Generic USB Device.
SubMsgID: 11, 00 SetKLDDigOutputs
DigOPs – 11, 00 I/O 1 and I/O 2 outputs set to ON (High).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	01	00	d	s

GET:
Response structure (12 bytes):
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
01	08	0C	00	d	s	SubMsgID		DigOPs		Reserved	

For structure see SET message above.

MGMSG_LA_SET_EEPROMPARAMS 0x0810

Function: Used to save the parameter settings for the specified message. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the ‘Settings’ button found in the lower right hand corner of the user interface).

SET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
10	08	02	00	d|	s	MsgID	

Data Structure:

field	description	format
MsgID	The message ID of the message containing the parameters to be saved.	word

Example:

TX 10, 08, 02, 00, D0, 01, 21, 08,

Header: 10, 08, 02, 00, D0, 01: Set_EEPROMPARAMS, 02 byte data packet, Generic USB Device.
MsgID: Save parameters specified by message 0821 (GetStatusUpdate).

MGMSG_LA_ENABLEOUTPUT 0x0811
MGMSG_LA_DISABLEOUTPUT 0x0812

Function These messages are sent to enable or disable the Laser output.
The 3rd and 4th bytes in the command header are unused and set to 0x00.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
11	08	00	00	d	s

Example: Enable the laser output

TX 11, 08, 00, 00, 50, 01

Disable the laser output

TX 12, 08, 00, 00, 50, 01

MGMSG_LD_OPENLOOP 0x0813
MGMSG_LD_CLOSEDLOOP 0x0814

These messages are applicable only to TLD001 Laser Diode Driver units

Function

The TLD001 laser diode driver can be operated in either Constant Current or Constant Power mode.
In OPEN LOOP or Constant Current Mode (CONST I), a constant drive current is applied to the laser diode. However, due to temperature fluctuations this does not result in a constant optical power output. As the diode warms up, the optical power will increase noticably from the level at initial switch on. Ambient temperature changes will also affect the output.
This mode is used when the lowest noise and highest response speed is required. Most applications in this mode will also require the temperature to be stabilized by an additional temperature controller. We offer the TTC001 TEC Controller T-Cube for such applications, see www.thorlabs.com for more details.

CLOSED LOOP or Constant Power Mode (CONST P) is used to minimize the output power fluctuations described above. This involves a signal from the internal photodiode, integrated into most laser diode packages, being fed back to the TLD001 unit in order to monitor and correct the power output.
An adjustment of the full scale photodiode current in CONST P mode is provided on the unit, in order to compensate for the differences in the photodiode currents between different laser diodes - see the manual supplied with the unit for more information on setting the photodiode current range.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	08	00	00	d	s

Example: Set the control mode to constant current (open loop)

TX 13, 08, 00, 00, 50, 01

Set the control mode to constant power (closed loop)

TX 14, 08, 00, 00, 50, 01

MGMSG_LD_POTROTATING 0x0815

This message is applicable only to TLD001 Laser Diode Driver units

Function This message is sent automatically by the system when the potentiometer on the TLD001 laser diode driver GUI panel is rotated by the user.
It contains the amount the pot has rotated (in degrees in bytes 2 and 3) since the last time the message was sent. This is represented as a signed short.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
15	08	00	00	d	s

MGMSG_LD_MAXCURRENTADJUST 0x0816

This message is applicable only to TLD001 Laser Diode Driver units

Function

In order to protect against damage which could be caused by operating errors, the limit for the Laser Diode drive current should be set before the diode is operated.
This message is called to enable and disable adjustment by setting byte 2 as follows:
Disable – 1
Enable - 2.
Note. When this message is called, the maximum current is reset to its minimum value (around 17mA). This ensures that initially, the laser current is at its lowest value.

Once Max Current Adjustment is enabled, the max current is set by calling the SET_MAXCURRENTDIGPOT message.

Byte 3 of the message is used to allow the current limit to be adjusted with the laser diode ON as follows:
Diode off - 1
Diode on - 2

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	08	00	00	d	s

Example: Set the unit to allow the laser diode max current to be adjusted with the output on

TX 13, 08, 02, 02, 50, 01

MGMSG_LD_SET_MAXCURRENTDIGPOT 0x0817
MGMSG_LD_REQ_MAXCURRENTDIGPOT 0x0818
MGMSG_LD_GET_MAXCURRENTDIGPOT 0x0819

This message is applicable only to TLD001 and KLD101 Laser Diode Driver units

Function

In order to protect against damage which could be caused by operating errors, the limit for the Laser Diode drive current should be set before the diode is operated.
Before calling this message, max current adjustment must be enabled by calling the MAXCURRENTADJUST message described previously. This message can then be called to set the max current for the laser diode being driven.
Note. When this message is called, the maximum current is reset to its minimum value (around 17mA for the TLD001 and 15 mA for the KLD101). This ensures that initially, the laser current is at its lowest value.

The max current is set in the range 0 to 255 which relates to 0 to 200 mA for the TLD001 or 230 mA for the KLD101.

SET:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
13	08	FF	00	d	s

Example: Set the max current to 200 mA

TX 13, 08, FF, 00, 50, 01

MGMSG_LD_FINDTIAGAIN 0x081A

This message is applicable only to TLD001 and KLD101 Laser Diode Driver units

Function

This message instructs the unit to find the optimum TIA gain setting for the TIA range currently selected.
Optimization of the TIA gain is an automated process performed internally by the unit, and should be performed only after the PD RANGE has been adjusted by setting the switches on the rear panel.
In the Thorlabs Software system, the software "demand" of how much current (in constant current mode) or optical power (in closed loop mode) is being generated by the laser diode is set by a digital to analog converter (DAC). This DAC produces a voltage that the software can set to be between zero and a fixed reference voltage.
When constant power mode is selected, a closed loop controller is set up that continuously reads the photocurrent and adjusts the laser power accordingly, so that the photocurrent is always equal to a "set point" value (the optical power is kept constant by keeping the photocurrent constant.).
To enable the full range of the DAC to be used, the photodiode current readings must be "normalized", so that the full range (i.e. maximum photocurrent) corresponds to the DAC full range. This normalization is performed when this message is called.

For example, assume the DAC generates a voltage between zero and 5 Volts maximum. In a particular set up, we may find that at maximum optical power, the photodiode produces 25 µA. When the message is called, the system adjusts the photodiode TIA gain to 0.2 V / µA so that the photodiode amplifier outputs 5 Volts.
In another setup, the photodiode produces a different current for max optical power, so a different photodiode amplifier gain is required.

Note. This message is sent automatically by the system once TIA Gain Adjustment is enabled by calling the LD_TIAGAINADJUST message.

Command structure (6 bytes):

0	1	2	3	4	5
header only					
1A	08	00	00	d	s

MGMSG_LD_TIAGAINADJUST 0x081B

This message is applicable only to TLD001 and KLD101 Laser Diode Driver units

Function This message is called to enable and disable TIA gain adjustment by setting byte 2 as follows:
Disable – 1
Enable - 2.
Once adjustment is enabled, the system sends the LD_FINDTIAGAIN message described previously to optimize the TIA gain for the range currently selected.

SET:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
1B	08	02	00	d	s

Example: Set the unit to allow the TIA gain to be adjusted

TX 1B, 08, 02, 00, 50, 01

MGMSG_LA_REQ_STATUSUPDATE 0x0820
MGMSG_LA_GET_STATUSUPDATE 0x0821

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the status of the controller (for example laser power or laser current). The response will be sent by the controller each time the function is requested.

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
20	08	00	00	d	s

GET:

Status update messages are received with the following format:-

Response structure (14 bytes)

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
21	08	08	00	d|	s	LaserCurrent		LaserPower		StatusBits			

Data Structure:

field	description	format
LaserCurrent	The laser current, in the range 0 to 32760 – (i.e. 0 to max current in mA)	word
LaserPower	The.laser power, in the range 0 to 32760 – (i.e. 0 to 100% of max power)	word
StatusBits	The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TLS001 Controller Bit Locations

Hex Value	Bit Number	Description
0x00000001	1	Laser output enabled state (1 - enabled, 0 - disabled).
0x00000002	2	Keyswitch enabled state (1 - enabled, 0 – disabled)
0x00000004	3	Laser control mode (1 - power [closed loop], 0 - current [open loop])
0x00000008	4	Safety interlock, (1 - enabled, 0 – disabled)
0x00000010	5	Units mode (1 - mA, else 0).
0x00000020	6	Units mode (1 - mW, else 0).
0x00000040	7	Units mode (1 - dBm, else 0)
	8 to 20	For Future Use

General Bit Locations

Hex Value	Bit Number	Description
0x00100000	21	Digital Input 1 (1 – logic high, 0 – logic low).
0x00200000	22	Digital Input 2 (1 – logic high, 0 – logic low).
0x40000000	31	Error

KLS101 Controller Bit Locations

Hex Value	Bit Number	Description
0x00000001	1	Laser output enabled state (1 - enabled, 0 - disabled).
0x00000002	2	Keyswitch enabled state (1 - enabled, 0 – disabled)
0x00000004	3	Laser control mode (1 - power [closed loop], 0 - current [open loop])
0x00000008	4	Safety interlock, (1 - enabled, 0 – disabled)
0x00000010	5 to 7	For Future Use
	8 to 19	Ext Input 12 bit ADC reading (1 LSB = 2.54mV, range 0 to 10.42V)

General Bit Locations

Hex Value	Bit Number	Description
0x00100000	20 to 30	For Future Use
0x00200000	31	Error (pigtail temperature > 50 °C)
0x40000000	31	Digital Feedback Settling

Example

RX 21, 08, 08, 00, 81, 50, 90, 19, 90, 19, 2B, 00, 00, 00

Header: 21, 08, 08, 00, 81, 50: LA_Get_StatusUpdate, 08 byte data packet, Generic USB Device.
LaserCurrent: 90, 19: 6544 = 20 % of the maximum current;
LaserPower: *90, 19*: 6544 = 20 % of the maximum power;
StatusBits: 2B,00,00,00, i.e. 00101011 the display shows mW units, the safety interlock is enabled, the keyswitch is enabled and the output is enabled.

MGMSG_LA_ACK_STATUSUPDATE 0x0822

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called “server alive” must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.status message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
22	08	00	00	d	s

TX 22, 08, 00, 00, 50, 01

MGMSG_LD_REQ_STATUSUPDATE 0x0825
MGMSG_LD_GET_STATUSUPDATE 0x0826

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the position and status of the controller (for example position and O/P voltage). The response will be sent by the controller each time the function is requested.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
20	08	00	00	d	s

GET:
Status update messages are received with the following format:-

Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
26	08	0E	00	d|	s	LaserCurrent		PhotoCurrent		LaserVoltage	

12	13	14	15	16	17	18	19
Data							
Reserved				StatusBits			

Data Structure:

field	description	format
LaserCurrent	The laser diode current, in the range -32768 to 32767 – (i.e. -200 to 200 mA)	word
PhotoCurrent	The photo diode current, in the range 0 to 32767 – (i.e. 0 to TIA Range Max in mA)	word
LaserVoltage	Laser Diode forward voltage -10000 to 10000 (-10.0V to 10.0V)	word
Reserved		dword
StatusBits	The meaning of the individual bits (flags) of the 32 bit integer value will depend on the controller and are described in the following tables.	dword

TLD001 controller Bit Locations

Hex Value	Bit Number	Description
0x00000001	1	Laser output enabled state (1 - enabled, 0 - disabled).
0x00000002	2	Keyswitch enabled state (1 - enabled, 0 – disabled)
0x00000004	3	Laser control mode (1 - power [closed loop], 0 - current [open loop])
0x00000008	4	Safety interlock, (1 - enabled, 0 – disabled)
0x00000010	5	TIA Range 1 (1 – 10µA, else 0).
0x00000020	6	TIA Range 2 (1 – 100µA, else 0).
0x00000040	7	TIA Range 3 (1 – 1 mA, else 0)
0x00000080	8	TIA Range 4 (1 – 10 mA, else 0)
0x00000100	9	Laser Diode Polarity (1 – Cathode Grounded, 0 – Anode Grounded)
0x00000200	10	External SMA Input Enabled (1 – Enabled, 0 – Disabled)
0x00000800	12	Laser Diode Open Circuit (1 – O/C, 0 – S/C)
0x00001000	13	All PSU Voltages OK (1 – OK, 0 – Not OK)
0x00002000	14	TIA Range Overlimit (1 – Overlimit, 0 – Not Overlimit)
0x00004000	15	TIA Range Underlimit (1 – Underlimit, 0 – Not Underlimit)

KLD101 controller Bit Locations

Hex Value	Bit Number	Description
0x00000001	1	Laser output enabled state (1 - enabled, 0 - disabled).
0x00000002	2	Keyswitch enabled state (1 - enabled, 0 – disabled)
0x00000004	3	Laser control mode (1 - power [closed loop], 0 - current [open loop])
0x00000008	4	Safety interlock, (1 - enabled, 0 – disabled)
0x00000010	5	TIA Range 1 (1 – 9µA, else 0).
0x00000020	6	TIA Range 2 (1 – 100µA, else 0).
0x00000040	7	TIA Range 3 (1 – 0.9 mA, else 0)
0x00000080	8	TIA Range 4 (1 – 10 mA, else 0)
0x00000100	9	Laser Diode Polarity (1 – Cathode Grounded, 0 – Anode Grounded)
0x00000200	10	External SMA Input Enabled (1 – Enabled, 0 – Disabled)
0x00000800	12	Laser Diode Open Circuit (1 – O/C, 0 – S/C)
0x00001000	13	All PSU Voltages OK (1 – OK, 0 – Not OK)
0x00002000	14	TIA Range Overlimit (1 – Overlimit, 0 – Not Overlimit)
0x00004000	15	TIA Range Underlimit (1 – Underlimit, 0 – Not Underlimit)

General Bit Locations

Hex Value	Bit Number	Description
0x00080000	20	Signal Generator ON (1 –YES, 0 – NO)
0x00100000	21	Digital Input 1 (1 – logic high, 0 – logic low).
0x00200000	22	Digital Input 2 (1 – logic high, 0 – logic low).
0x40000000	31	Error
0x80000000	32	High stability reached (1 –YES, 0 – NO)

MGMSG_LD_ACK_STATUSUPDATE 0x0827

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called "server alive" must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
27	08	00	00	d	s

TX 27, 08, 00, 00, 50, 01

MGMSG_LA_SET_KCUBETRIGIOCONFIG 0x082A
MGMSG_LA_REQ_KCUBETRIGCONFIG 0x082B
MGMSG_LA_GET_KCUBETRIGCONFIG 0x082C

This message is applicable only to KLS635 and KLS1550 units

Function: The K-Cube laser source units have two bidirectional trigger ports (TRIG1 and TRIG2) that can be used to read an external logic signal or output a logic level to control external equipment. Either of them can be independently configured as an input or an output and the active logic state can be selected High or Low to suit the requirements of the application. Electrically the ports output 5 Volt logic signals and are designed to be driven from a 5 Volt logic.
When the port is used in the input mode, the logic levels are TTL compatible, i.e. a voltage level less than 0.8 Volt will be recognised as a logic LOW and a level greater than 2.4 Volt as a logic HIGH. The input contains a weak pull-up, so the state of the input with nothing connected will default to a logic HIGH. The weak pull-up feature allows a passive device, such as a mechanical switch to be connected directly to the input.
When the port is used as an output it provides a push-pu
ll drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuity.
Warning: do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

SET
Command structure (20 bytes)
6 byte header followed by 14 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
2A	08	0C	00	d	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19
Data							
Reserved		Trig2Mode		Trig2Polarity		Reserved	

Data Structure:

field	description	format
Chan Ident	The channel being addressed is always encoded as a 16-bit word (0x01 0x00)	word
Trig1Mode	TRIG1 operating mode	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) l.	word
Reserved		
Trig2Mode	TRIG2 operating mode	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low)	word
Reserved		

Input Trigger Modes
When configured as an input, the TRIG ports can be used as a general purpose digital input, or for triggering a choice of actions as follows:

0x00 The trigger IO is disabled
0x01 General purpose logic input (read through status bits using the LA_GET_STATUSUPDATE message or the Get Status Bits sub message of the LA_GET_PARAMS message).

When used for triggering, the port is edge sensitive. In other words, it has to see a transition from the inactive to the active logic state (Low->High or High->Low) for the trigger input to be recognized. For the same reason a sustained logic level will not result in repeated trigger signals. The trigger input has to return to its inactive state first in order to start the next trigger.

Output Trigger Modes
When configured as an output, the TRIG ports can be used as a general purpose digital output, or to indicate status or to produce a trigger pulse at configurable events as follows:

0x0A General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).
0x0B Trigger output active when the laser output is ON. The output trigger goes high (5V) or low (0V) (as set in the Polarity parameter) when the laser is active.
0x0C Trigger output active when the interlock state is Enabled
0x0D Trigger output active when the laser set point value is changed. (pulse signal)

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
2B	08	Chan Ident	00	d	s

Example: Request the Trigger IO settings

TX 2B, 08, 01, 00, 50, 01

GET:

Response structure (18 bytes):
6 byte header followed by 12 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
2C	08	0C	00	d	s	Chan Ident		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19
Data							
Trig1Par		Trig2Mode		Trig2Polarity		Trig2Par	

For structure see SET message above.

Quad Control Messages

Introduction

The ‘Quad’ ActiveX Control provides the functionality required for a client application to control one or more T-Cube Quad Detector Readers or Position Aligners.
The methods of the Quad Control Object can then be used to control the TQD001 T-Cube Quad Reader, the TPA101 T-Cube Position Aligner and the KPA101 K-Cube Position Aligner, to perform activities such as switching between Monitor, Open Loop and Closed Loop operating modes, setting the position demand parameters, reading the present beam position and setting the LED display intensity.
For details on the use of the T-Cubes and K-Cube, refer to the handbook supplied for the unit.

MGMSG_QUAD_SET_PARAMS 0x0870
MGMSG_QUAD_REQ_PARAMS 0x0871
MGMSG_QUAD_GET_PARAMS 0x0872

Function: This generic parameter set/request message is used to control the functionality of the TQD001, TPA101 and KPA101 units. The specific parameters to control are identified by the use of sub-messages. These sub messages comply with the general format of the Thorlabs message protocol but rather than having a unique first and second byte in the header carrying the "message identifier" information, the first and second byte remain the same.
Instead, for the SET and GET messages, the message identifier is carried in the first two bytes in the data packet part of the message, whilst for the REQ message it is encoded as the third byte of the header.
Likewise, when the unit responds, the first two bytes of the response remain the same and the first two bytes of the data packet identify the sub-message to which the information returned in the remaining part of the data packet relates.

The following sub messages are applicable to the TQD001, TPA101 and KPA101:

Set/Request/Get Quad_LoopParams (sub-message ID = 01)
Request/Get Quad_Readings (sub-message ID = 03)
Set/Request/Get Quad Position Demand Params (sub-message ID = 05)
Set/Request/Get Quad Operating Mode (sub-message ID = 07)
Request/Get Quad Status Bits (sub-message ID = 09)
Set/Request/Get Quad Display Settings (sub-message ID = 0B)
Set/Request/Get Quad Position Demand Outputs (sub-message ID = 0D)

The following sub message is applicable only to the TPA101 and KPA101:

Set/Request/Get Quad_LoopParams2 (sub-message ID = 0E)

To explain the principle, the following examples describe these messages in more detail.

Set/Request/Get Quad_LoopParams (sub-message ID = 01)
Used to set the proportional, integration and derivative feedback loop constants to the value specified in the PGain, IGain and DGain parameters respectively. They apply when the quad detector unit is operated in closed loop mode, and position demand signals are generated at the rear panel SMA connectors by the feedback loops. These position demand voltages act to move the beam steering elements (e.g. a piezo driven mirror) in order to centralize a beam at the centre of the PSD head.
When operating in closed loop mode, the proportional, integral and derivative (PID) constants can be used to fine tune the behaviour of the dual feedback loops to adjust the response of the position demand output voltages. The feedback loop parameters need to be adjusted to suit the different types of sensor that can be connected to the system. The default values have been optimized for the PDQ80A sensor.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
70	08	08	00	d|	s	SubMsgID		PGain		IGain		DGain	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0100) of the message containing the parameters	word
PGain	The proportional gain. This term provides the force used to drive the piezo to the demand position, reducing the positional error. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767 (i.e. 0 to 100 in Thorlabs User GUI).	word
IGain	The integral gain. This term provides the 'restoring' force that grows with time, ensuring that the positional error is eventually reduced to zero. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 32767 (i.e. 0 to 100 in Thorlabs User GUI).	word
DGain	The derivative gain. This term provides the 'damping' force proportional to the rate of change of the position. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 32767 (i.e. 0 to 100 in Thorlabs User GUI).	word

Example: Set the PID parameters for TQD001 or TPA101 as follows:
Proportional: 65
Integral: 80
Derivative: 60

TX 70, 08, 08, 00, D0, 01, 01, 00, 41, 00, 50, 00, 3C, 00,

Header: 70, 08, 08, 00, D0, 01: Quad_SetParams, 8 byte data packet, Generic USB Device.
SubMsgID: 01, 00 SetQuadControlLoopParams)
PGain: *32, 53,(32767x65/100):* Set the proportional term to 65
IGain: *65, 66, (32767x80/100)*: Set the integral term to 80
DGain: CC, 4C, *(32767x60/100)*: Set the derivative term to 60

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	01	00	d	s

GET:

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
72	08	08	00	d|	s	SubMsgID		PGain		IGain		DGain	

For structure see Set message above.

Request/Get Quad_Readings (sub-message ID = 3)

The TQD001, TPA101 and KPA101 control units have been designed to operate with the PDQ80A and PDQ30C Quad Detectors and the PDP90A Lateral Effect Position Sensor. These detectors consist of a 4-segment photodiode sensor array, which provides 'Bottom minus Top' (YDIFF) and 'Left minus Right' (XDIFF) difference signals, together with the SUM of the signals (total beam power) from all four quadrants of the photodiode array.

This sub-message is used to read the actual SUM, XDIFF and YDIFF signals from the detector. Whether these signals are routed to the LV OUT/XDIFF and LV OUT/YDIFF SMA connectors on the rear panel depends on the operating mode selected (see the Quad_OperMode message) as follows.

In 'Closed Loop' mode, the signal from the detector is interpreted by the unit, and the feedback circuit sends position demand signals (XOut and YOut) to the rear panel LV OUT/XDIFF and LV OUT/YDIFF connectors, which can be used to drive a pair of positioning elements (e.g. piezo controllers) in order to position the light beam within the center of the detector array. This submessage is then used to read the actual values for the XPos and YPos position demand signals (-10 V to +10V). Note that in closed loop mode, with the beam central, the X and Y axis difference outputs from the photodiode array are zero. However, the position demand signals on the rear panel LV OUT XDIFF and YDIFF SMA connectors are whatever value is necessary to drive the positioning elements to centre the beam.

When the unit is operated in 'open loop' mode, the signals on the rear panel XDIFF and YDIFF connectors are constant. They are either fixed at zero (0V), or held at the last Closed Loop value (depending on the 'QuadPosDemandParams' message. This is useful when the system is being adjusted manually, to position the light beam within the detector array.

When operating in 'Monitor' mode, the X axis (XDIFF) and Y axis (YDIFF) difference signals from the detector, are fed through to the rear panel SMA connectors for use in a monitoring application.

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	03	00	d	s

TX 71, 08, 03, 00, 50, 01,

GET:

Command structure (18 bytes)

6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	08	0C	00	d|	s	SubMsgID		XDiff		YDiff	

12	13	14	15	16	17
Data					
Sum		XPos		YPos	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0300) of the message containing the parameters	word
XDiff	The present X axis difference (XDIFF) signal from the detector head. (-10V to 10V in the range -32768 to 32767)	short
YDiff	The present Y axis difference (YDIFF) signal value from the detector head. (-10V to 10V in the range -32768 to 32767)	short
Sum	The present Sum signal value from the detector head (0V to 10V in the range 0 to 65535)	word
XPos	The X axis position output value on the rear panel XDiff SMA connector (-10V to 10V in the range -32768 to 32767)	short
YPos	The Y axis position output value on the rear panel YDiff SMA connector (-10V to 10V in the range -32768 to 32767)	short

Example: Get the Quad Detector T-Cube readings (T-Cube in open loop mode)

RX 72, 08, 0C, 00, D0, 01, 03, 00, FF, 3F, FF, 3F, FF, 7F, 00, 00, 00, 00

Header: 72, 08, 0C, 00, D0, 01: Quad_GetPARAMS, 12 byte data packet, Generic USB Device.
MsgID: 03, 00: Get Quad Readings
XDiff:.FF, 3F: 0x3FFF (16383 decimal), i.e. 5 V.
YDiff:. FF, 3F: 0x3FFF (16383 decimal), i.e. 5 V.
Sum: FF, FF: 0x7FFF (65535 decimal), i.e. 10 V.
XPos: 00, 00 i.e. Zero
YPos: 00, 00 i.e. Zero

Set/Request/Get Quad_PosDemandParams (sub-message ID = 5)

The TQD001, TPA101 and KPA101 control units have been designed to operate with the PDQ80A and PDQ30C Quad Detectors and the PDP90A Lateral Effect Position Sensor. These detectors consist of a 4-segment photodiode sensor array, which provides 'Bottom minus Top' (YDIFF) and 'Left minus Right' (XDIFF) difference signals, together with the SUM of the signals (total beam power) from all four quadrants of the photodiode array. Whether these signals are routed to the LV OUT/XDIFF and LV OUT/YDIFF SMA connectors on the rear panel depends on the operating mode selected – see the Quad_OperMode message.

This sub-message is used to control the signals on the rear panel LV OUT/XDIFF and LV OUT/YDIFF connectors.

SET:

Command structure (24 bytes)

6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
70	08	12	00	d|	s	SubMsgID		XPosDemMin		YPosDemMin	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
XPosDemMax		YPosDemMax		LVOutRoute		OLPosDem		XPosFBSense		YPosFBSense	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0500) of the message containing the parameters	word
XPosDemandMin	The following four parameters are applicable only when operating in closed loop mode. The XOut and YOut values are the low voltage signals sent to the LV OUT/XDIFF and LV OUT/YDIFF connectors, which are then used to drive the positioning mechanism in order to keep the beam central in the detector. Under normal operating conditions, these values are between -10 V and +10 V, however some applications may require the limits to be less than this. The XPosDemandMin parameter is used to set the min limit for the XOut value, between -10V and +10V. (i.e. -32768 to 32767)	short
YPosDemandMin	As above. The YPosDemandMin parameter is used to set the min limit for the YOut value, between -10V and +10V. (i.e. -32768 to 32767)	short
XPosDemandMax	As above. The XPosDemandMax parameter is used to set the max limit for the XOut value, between -10V and +10V. (-32768 to 32767)	short
YPosDemandMax	As above. The YPosDemandMax parameter is used to set the max limit for the YOut value, between -10V and +10V. (-32768 to 32767)	short
LVOutRoute	When operating in closed loop mode, the Quad Detector position control signals are always output on the external SMA connectors (LV OUT XDiff and LV	word

	OUT YDiff). In addition, they can also be routed to the TCH002 hub, which eliminates the need for external SMA to SMA cables. This parameter is used to set the LV Out signal routing as follows: 1 SMA Only 2 SMA + Hub	
OpenLoopPosDemands	When the Quad Detector T-Cube is operated in 'open loop' mode, the position demand signals (on the XDIFF and YDIFF connectors) can either be set to zero, or held at their last closed loop value, according to the value entered in this parameter as follows: 1 OpenLoopPosDemandsZero - the output is set to zero (0V). 2 OpenLoopPosDemandsHeld = the outputs are fixed at the values present when the unit is switched to open loop.	word
XPosDemandFBSense	Due to the choice of piezo amplifier/driver or the configuration of mirrors (or other optical components) it is possible that certain application set ups may require the sense of the X and Y axis position demand signals to be inverted. This parameter sets the signal sense and gain for the X axis output as follows: If XPosDemandFBSense is set to '10' (32767) the signals are positive when the beam is in the left hand quadrants of the detector array, and negative when in the right hand quadrants. The gain of the system is set to '1'. If XPosDemandFBSense is set to '-7' (-22938) the signals are positive when the beam is in the right hand quadrants of the detector array, and negative when in the left hand quadrants. The gain of the system is set to '0.7'.	short
YPosDemandFBSense	Similarly to the XPosDemandFBSense described above, this parameter sets the signal sense and gain for the Y axis output as follows: If YPosDemandFBSense is set to '10' (32767) the signals are positive when the beam is in the top quadrants of the detector array, and negative when in the bottom quadrants. The gain of the system is set to '1'. If YPosDemandFBSense is set to '-3' (-9830) the signals are positive when the beam is in the bottom quadrants of the detector array, and negative when in the top quadrants. The gain of the system is set to '0.3'.	short

Example: Set the Quad Pos Demand Params

RX 70, 08, 12, 00, D0, 01, 05, 00, 01, 80, 01, 80, FF, 7F, FF, 7F, 02, 00, 01, 00, 0A, 00, 0A, 00

Header: 70, 08, 12, 00, D0, 01: Quad_SetPARAMS, 18 byte data packet, Generic USB Device.
SubMsgID: 05, 00: Set Quad PosDemandParams
XPosDemandMin:.01, 80: 0x8001 (-32767 decimal), i.e. -10 V.
YPosDemandMin:. 01, 80: 0x8001 (-32767 decimal), i.e. -10 V.
XPosDemandMax: FF, 7F: 0x7FFF (32767 decimal), i.e. 10 V.
YPosDemandMax: FF, 7F: 0x7FFF (32767 decimal), i.e. 10 V.
LVOutRoute: 02, 00 i.e. SMA + Hub
OpenLoopPosDemand:.01, 00: i.e. Zero.
XPosDemandFBSense:. FF, 7F: i.e. Positive sense, gain = 1.
YPosDemandFBSense: 9A, D9: i.e. Positive sense, gain = 0.3.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	05	00	d	s

TX 71, 08, 05, 00, 50, 01,

GET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	08	12	00	d|	s	SubMsgID		XPosDemMin		YPosDemMin	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
XPosDemMax		YPosDemMax		LVOutRoute		OLPosDem		XPosFBSense		YPosFBSense	

See Set message for structure

Set/Request/Get Quad_OperMode (sub-message ID = 07)
Used to set the operating mode of the control unit to either Monitor, Open Loop or Closed Loop mode as described below.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
70	08	08	00	d|	s	SubMsgID		Mode	

Data Structure:

field	description	format
SubMsg ID	The message ID (i.e. 0700) of the message containing the parameters	word
Mode	The operating mode of the unit. When operating in 'Monitor' mode, the X axis (XDIFF) and Y axis (YDIFF) difference signals from the detector, are fed through to the rear panel SMA connectors for use in a monitoring application. When in 'Open Loop' mode, the signals at the rear panel are fixed at zero (0V), or held at the last closed loop value, depending on the setting of the 'OpenLoopPosDemands parameter in the QuadPosDemandParams message. This is useful when the system is being adjusted manually, to position the light beam within the detector array. In 'Closed Loop' mode, the feedback circuit sends position demand signals to the rear panel XDIFF and YDIFF connectors, which can be used to drive a pair of positioning elements (e.g. piezo drivers) in order to position the light beam within the center of the detector array. The mode is set as follows: 1 Monitor Mode 2 OpenLoop 3 ClosedLoop The following mode is applicable only to the KPA101 K-Cube Position Aligner 4 Auto Open/Closed Loop Mode: the unit operates in closed loop' mode, until the SUM signal falls below the value set in the SumMin parameter of the Set/Request/Get Quad_KPATRIGIOCONFIG method.	word

A Note About Automatic Open Loop/Closed Loop Switching

The KPA101 controller is capable of switching automatically between open loop and closed loop operating modes, depending on whether there is sufficient optical power required for

closed loop operation. Automatic Switching mode can be selected by setting the Mode parameter to 4_AUTOOPENCLOSEDLOOP as described above.
If during closed loop operation the SUM signal falls below the minimum specified in the SumMin parameter of the Set/Request/Get Quad_KPATRIGIOCONFIG method, the controller will switch back to open loop mode. If subsequently the SUM signal rises above the limit again, the controller will switch back to closed loop mode.
The automatic switchover works in conjunction with the "Position Demands In Open Loop Mode" option in the SetQuad_PosDemandParams submessage, that defines whether the controller will hold (freeze) the XPOS and YPOS outputs when switching over to open loop or set them to zero.
Automatic switchover might be advantageous in scenarios where the beam might be temporarily blocked, for example during experiments involving manual manipulation of optical components, particularly when the beam path is quite long and the beam steering actuator can deflect the beam so far that it falls outside the sensor area. In setups like this and with the controller in closed loop, blocking the beam can result in the feedback loop ramping the XPOS and/or YPOS outputs to saturation and steering the beam completely outside the sensor area. When this happens, restoring the beam will not normally restore the beam alignment as at this point the feedback algorithm does not even see the beam. However, with automatic switchover the loss of light will stop the closed loop operation, optionally freeze the last valid beam position and prevent the outputs ramping up as an unintentional consequence of the loss of feedback signals. Later when the beam is restored, closed loop operation will resume and continue control starting from the last valid beam position.
Note that because automatic switchover assumes the knowledge of the last valid closed loop beam position that is lost when the controller is powered down, this option cannot be persisted. For a similar reason, the controller will always power up in open loop mode.
Example: Set the operating mode to closed loop

TX 70, 08, 04, 00, D0, 01, 07, 00, 03, 00,

Header: 70, 08, 04, 00, D0, 01: Quad_SetPARAMS, 04 byte data packet, Generic USB Device.
SubMsgID: 07, 00: SetQuadOperMode
Mode: *03, 00,:* Set closed loop mode

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	Msg Ident	00	d	s

GET:
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
70	08	08	00	d|	s	SubMsgID		Mode	

For structure see Set message above.

Request/Get Quad_Status Bits (sub-message ID = 9)
This sub command can be used to request the control unit status bits. The message only has a request/ get part.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	09	00	d	s

TX 71, 08, 09, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	08	06	00	d|	s	SubMsgID		StatusBits			

Data Structure:

field	description	format
MsgID	The message ID (0900) of the message containing the parameters	word
StatusBits	The individual bits (flags) of the 32 bit integer value are described in the following table.	dword

TQD001 or TPA101 controller

Hex Value	Bit Number	Description
0x00000001	1	Position Monitoring Mode (1 - enabled, 0 - disabled).
0x00000002	2	Open Loop Operating Mode (1 - enabled, 0 – disabled)
0x00000004	3	Closed Loop Operating Mode (1 - enabled, 0 – disabled)
0x00000008	4 to 32	For Future Use

Example

RX 72, 08, 06, 00, D0, 50, 09, 00, 2B, 00, 00, 00

Header: 02, 08, 06, 00, D0, 50: Quad_Get_Params, 06 byte data packet, Generic USB Device.
MsgID: 09, 00: Get Status Bits
StatusBits: 04,00,00,00, i.e. 100 Closed Loop operating mode is enabled.

Set/Request/Get Quad Display Settings (sub-message ID = 0B)

This message can be used to adjust or read the front panel LED display brightness and the display units.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
70	08	08	00	d|	s	SubMsgID		DispIntensity		DispMode		DispDimTimeout	

Data Structure:

field	description	format
MsgID	The message ID (i.e. 0B00) of the message containing the parameters	word
DispIntensity	The intensity is set as a value from 0 (Off) to 255 (brightest).	word
DispMode	The main display on the GUI panel can be set to show X and Y axis difference signals from the detector array (Difference) or the Xpos and Ypos position demand output signals fed to the positioning elements (Position) as follows: 1 QUAD_DISPMODE_DIFF, the display represents the X and Y axis difference signals from the detector (i.e. the voltage outputs from the rear panel SMA connectors in Monitor Mode). 2 QUAD_DISPMODE_POS, the display represents the position of the XPos and YPos position demand output signals fed to the positioning elements (i.e. the voltage outputs from the rear panel SMA connectors in OPEN or CLOSED loop mode).	word
DispDimTimeout	'Burn In' of the display can occur if it remains static for a long time. To prevent this, the display is automatically dimmed after a specified time interval has elapsed. The brightness level after dimming is set as a percentage of full brightness, from 0 (Off) to 10 (brightest). The values are passed in the form (512 x DimLevel) + Timeout – see example below.	word

Example: Set the display to max brightness, the display mode to Difference, the timeout to 10 minutes and the dim level to 5.

TX 70, 08, 08, 00, D0, 01, 0B, 00, FF, 00, 01, 00, 0A, 0A

Header: 70, 08, 08, 00, D0, 01: Quad_SetParams, 08 byte data packet, Generic USB Device.
SubMsgID: 0B, 00: Set Display Settings
DispIntensity: FF, 00: Sets the display brightness to 255 (100%)
DispMode: 01, 00: Sets the display mode to option 1, i.e. Difference
DispDimTimeout: 0A, 0A: Sets the DispDimTimeout parameter to 2570, which equates to a 2570/512 = 5, with a timeout of 10 minutes

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	0B	00	d	s

Example: TX 71, 08, 0B, 00, 50, 01

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
72	08	08	00	d|	s	SubMsgID		DispIntensity		DispMode		DispDimTimeout	

See SET for data structure.

Set/Request/Get Quad_PositionOutputs (sub-message ID = 0D)
This sub message can be used to set and get the position demand signals (on the XDIFF, YDIFF connectors).
When the quad detector unit is used with a beam steering device (e.g. a piezo mirror via piezo drivers), this message allows the beam to be positioned by entering a value (-10 V to +10V) in the XPos and YPos parameters.

SET:
Status update messages are received with the following format:-

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
70	08	06	00	d|	s	SubMsgID		XPos		YPos	

Data Structure:

field	description	format
MsgID	The message ID (i.e. 0D00) of the message containing the parameters	word
XPos	The X axis position output value -10 V to 10 V (i.e. -32768 to 32767)	short
YPos	The Y axis position output value -10 V to 10 V (i.e. -32768 to 32767)	short

Example Set the XPos and YPos signals to be -10 V and 10V respectively.

TX 70, 08, 06, 00, D0, 01, 0D, 00, 01, 80, FF, 7F

Header: 70, 08, 06, 00, D0, 01: Quad_Get_Params, 06 byte data packet, Generic USB Device.
MsgID: 0D, 00: Get Quad_PositionOutputs
XPos:. 01, 80: 0x8001 (-32767 decimal), i.e. -10 V.
YPos: FF, 7F: 0x7FFF (32767 decimal), i.e. 10 V.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	0D	00	d	s

TX 71, 08, 0D, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	08	06	00	d|	s	SubMsgID		XPos		YPos	

Set/Request/Get Quad_LoopParams2 (sub-message ID = 0E)

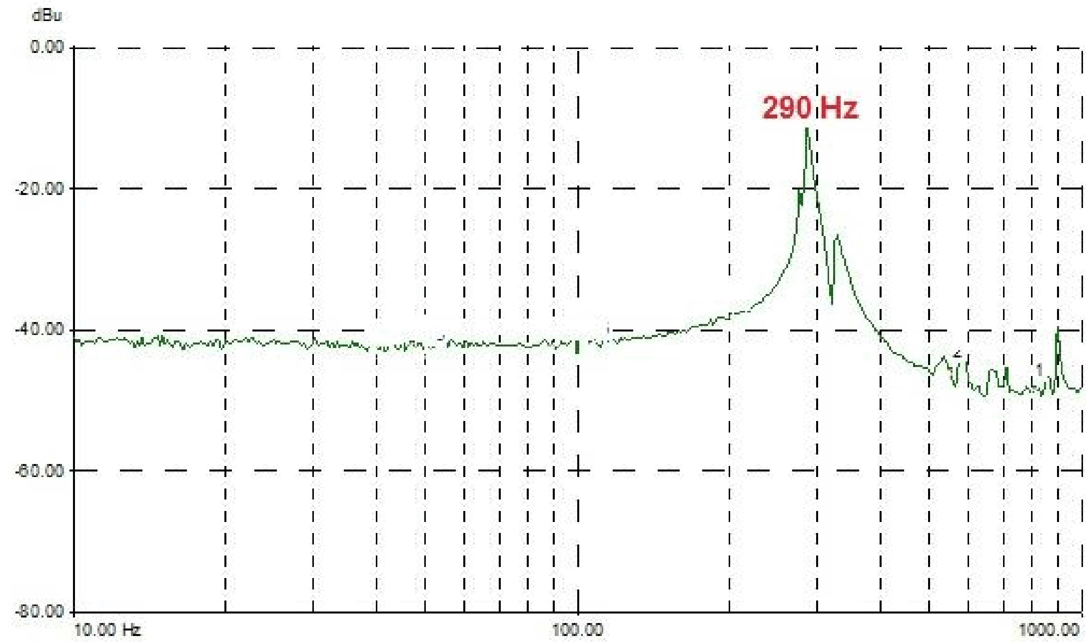
This sub-message is applicable only to the TPA101 and KPA101 units.

Used to set the proportional, integration and derivative feedback loop constants and also to set the derivative cut off frequency and the notch filter center frequency.

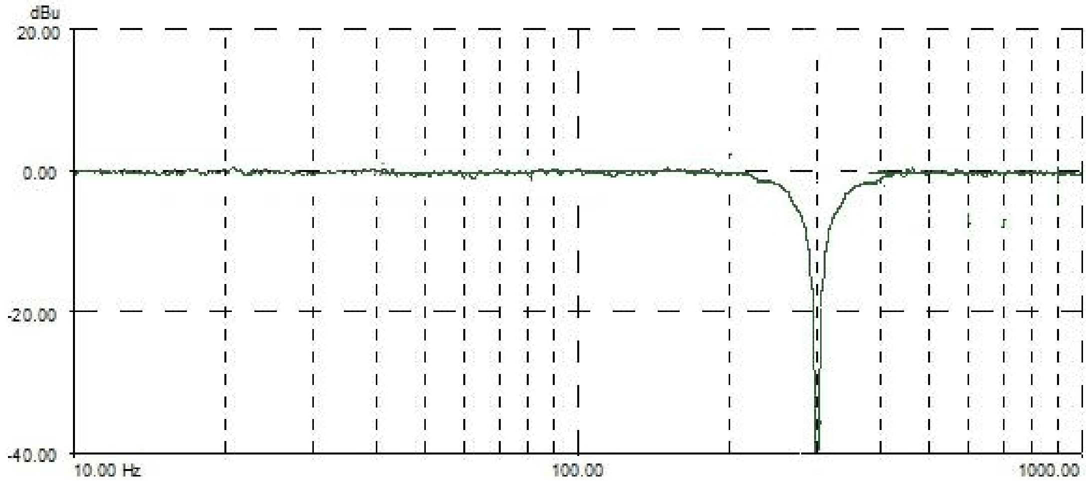
PID Constants: The PID constants apply when the unit is operated in closed loop mode, and position demand signals are generated at the rear panel SMA connectors by the feedback loops. These position demand voltages act to move the beam steering elements (e.g. a piezo driven mirror) in order to centralize a beam at the centre of the PSD head.
When operating in closed loop mode, the proportional, integral and derivative (PID) constants can be used to fine tune the behaviour of the dual feedback loops to adjust the response of the position demand output voltages. The feedback loop parameters need to be adjusted to suit the different types of sensor that can be connected to the system. The default values have been optimized for the PDQ80A sensor.

*Derivative Filte*r: The output of the derivative part of the PID controller can be passed through a tuneable low pass filter. Whilst the derivative component of the PID loop often improves stability (as it acts as a retaining force against abrupt changes in the system), it is prone to amplifying noise present in the system, as the derivative component is sensitive to changes between adjacent samples. To reduce this effect, a low pass filter can be applied to the samples. As noise often tends to contain predominantly high frequency components, the low pass filter can significantly decrease their contribution, often without diminishing the beneficial, stabilizing effect of the derivative action. In some applications enabling this filter can improve the overall closed loop performance.

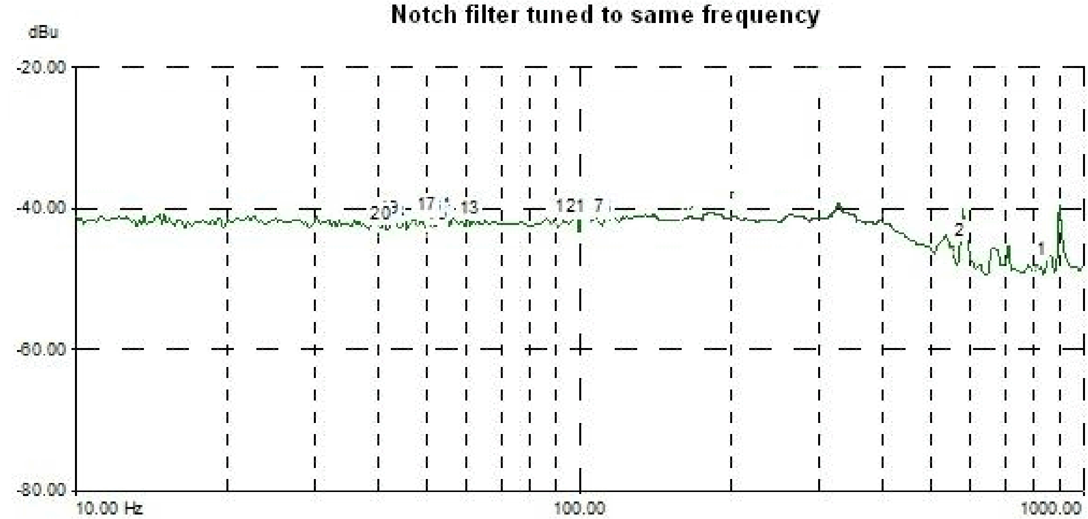
Notch Filter: Due to their construction, most actuators are prone to mechanical resonance at well-defined frequencies. The underlying reason is that all spring-mass systems are natural harmonic oscillators. This proneness to resonance can be a problem in closed loop systems because, coupled with the effect of the feedback, it can result in oscillations. With some actuators (for example the ASM003), the resonance peak is either weak enough or at a high enough frequency for the resonance not to be troublesome. With other actuators (for example the PGM100) the resonance peak is very significant and needs to be eliminated for operation in a stable closed loop system. The notch filter is an adjustable electronic anti-resonance that can be used to counteract the natural resonance of the mechanical system. As the resonance frequency of actuators varies with load in addition to the minor variations from product to product, the notch filter is tuneable so that its characteristics can be adjusted to match those of the actuator. In addition to its centre frequency, the bandwidth of the notch (or the equivalent quality factor, often referred to as the Q-factor) can also be adjusted. In simple terms, the Q factor is the centre frequency/bandwidth, and defines how wide the notch is, a higher Q factor defining a narrower ("higher quality") notch. Optimizing the Q factor requires some experimentation but in general a value of 5 to 10 is in most cases a good starting point.



Frequency response of actuator showing resonance at 290 Hz



Notch filter tuned to same frequency



The resonance is largely eliminated

SET:
Command structure (36 bytes)
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
70	08	1E	00	d|	s	SubMsgID		PIDConstsP				PIDConstsI	

14	15	16	17	18	19	20	21	22	23	24	25	26	27
Data													
PIDConstsI		PIDConstsD				PIDConstsDFc				FilterFc			

28	29	30	31	32	33	34	35
Data							
FilterQ				NotchFilterOn		PIDDerivFilterOn	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0E,00) of the message containing the parameters	word
PIDConstsP	The proportional gain. This term provides the force used to drive the piezo to the demand position, reducing the positional error. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 10000.	float
PIDConstsI	The integral gain. This term provides the ‘restoring’ force that grows with time, ensuring that the positional error is eventually reduced to zero. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 10000.	float
PIDConstsD	The derivative gain. This term provides the ‘damping’ force proportional to the rate of change of the position. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 10000.	float
PIDConstsDFc	The cut off frequency of the Derivative Low Pass Filter, in the range 0 to 10,000	float
FilterFc	The Notch Filter center frequency, in the range 0 to 10,000	float
FilterQ	The Notch Filter Q factor, in the range 0.1 to 100	float
NotchFilterOn	Turns the notch filter on (set to 1) and off (set to 2)	word
PIDDerivFilterOn	Turns the derivative filter on (set to 1) and off (set to 2)	word

Example: Set the PID parameters for TPA101 as follows:
Proportional: 65.7
Integral: 80.3
Derivative: 60.9
Derivative LP Cutoff: 500 Hz
Notch Filter Center Freq: 500Hz
Q Factor: 5.0
Notch Filter ON
Derivative Filter ON

TX 70, 08, 1E, 00, D0, 01, 0E, 00, 66, 66, 83, 42, 9A, 99, A0, 42, 9A, 99, 73, 42, 00, 00, FA, 43, 00, 00, FA, 43, 00, 00, A0, 40, 01, 00, 01, 00

Header: 70, 08, 1E, 00, D0, 01: Quad_SetParams, 30 byte data packet, Generic USB Device.
SubMsgID: 0E, 00 SetQuadControlLoopParams2)
Prop: *66, 66, 83, 42:* Set the proportional term to 65.7
Int: *9A, 99, A0, 42*: Set the integral term to 80.3
Deriv: 9A, 99, 73, 42: Set the derivative term to 60.9
Derivative LP Cut Off: 00, 00, FA, 43: Set the low pass cut off frequency to 500 Hz
Notch Filter Center: 00, 00, FA, 43: Set the notch filter center frequency to 500 Hz
Q Factor: 00, 00, A0, 40: Set the Q factor to 5.0
Notch Filter ON: 01, 00: Set the notch filter ON
Derivative Filter ON: 01, 00: Set the low pas filter ON.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	01	00	d	s

GET:
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
72	08	1E	00	d|	s	SubMsgID		PIDConstsP				PIDConstsI	

14	15	16	17	18	19	20	21	22	23	24	25	26	27
Data													
PIDConstsI		PIDConstsD				PIDConstsDFc				FilterFc			

28	29	30	31	32	33	34	35
Data							
FilterQ				NotchFilterOn		PIDDerivFilterOn	

For structure see Set message above.

Set/Request/Get Quad_KPATRIGIOCONFIG (sub-message ID = 0F)

This sub-message is applicable only to KPA101 units.

Used to set the operating parameters of the trigger connectors on the front panel of the unit.
The K-Cube position aligner has two bidirectional trigger ports (TRIG1 and TRIG2) that can be independently configured either as an input or an output and assigned a function from the list of options described in the following section. The polarity (logic HIGH / LOW or rising / falling edge) can also be configured to suit the requirements of the equipment connected to these ports.
In the input operating modes the port is electrically configured as a TTL compatible logic input. When the port is driven with a voltage level below +0.8 V, it will read a logic LOW and when driven above +2.4V, it will read a logic HIGH. The ports have an internal weak pull-up resistor ensuring that a stable logic level is present on the inputs even when there is no driving source connected to it. This means that when unconnected the ports will read a logic HIGH. The internal pull-up also allows the direct connection of mechanical switches or other unpowered control devices.
In the output modes the port is electrically configured as a logic output using 5 Volt logic levels. The port is connected to the output driver logic with a 620 Ohm resistor in series; this resistor limits the maximum output current to approximately 8 mA and provides protection against the output being accidental short circuited to ground. The output can be used to drive the majority of digital inputs used on external equipment without any additional circuitry.

Warning: do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

Trigger Modes
Input Trigger Modes
0x00 TRIG_DISABLED - The trigger IO is disabled. Selecting this option effectively results in the port returning to its default digital input configuration

0x01 TRIGIN_GPI - General purpose logic input. Other than being able to read the logic state of port there is no other functionality associated with it. The state of the port is returned in the periodic status update messages, or can be read by using the Get_Quad_Status Bits sub-message). In this mode the Triggering Polarity setting has no effect; the logic state of the input is returned as it is present on the port without inversion.

0x02 TRIGIN_LOOPOPENCLOSE - In this mode the port can be used to toggle the operating mode of the controller between open loop and closed loop modes. If the trigger polarity is selected to be "Active High", the operating mode toggles on the rising edge (LOW to HIGH transition) of the signal present on the TRIG input. Conversely, with "Active Low", the toggle takes place on the falling edge (HIGH to LOW transition).

Output Trigger Modes
0x0A TRIGOUT_GPO - In this operating mode the TRIG port functions as a simple digital output. The logic state of the output can be set using the MOD_SET_DIGOUTPUTS message. Other than being able to read the logic state of port there is no other functionality

associated with it. The logic state of the output can be inverted by setting the Triggering Polarity parameter to "Low"; with this option selected the state of the output will be the opposite of the corresponding bit setting in the software call. The default state of the output in this mode is also the opposite of the option selected as the Triggering Polarity.

0x0B TRIGOUT_SUM - The state of the TRIG port is asserted depending on whether the SUM signal coming from the position sensor is inside the limits specified in the lSumMin and lSumMax parameters. If SUM is within the limits, the state will be the logic state selected in Triggering Polarity and conversely if it falls outside these limits, it will assume the opposite logic state. This mode can be used to detect the presence or absence of light falling on the position sensor; or that the optical power is within the expected limits. This option might be useful to signal a condition required for normal operation as under normal operating conditions the optical power is often expected to remain fairly constant. The lSumMin and lSumMax parameters are specified as a percentage of full scale, in the range 1% to 99%.

0x0C TRIGOUT_DIFF - The state of the TRIG port is asserted depending on whether both the XDIFF and the YDIFF signals coming from the position sensor are below the value set in the lDiffThreshold parameter. If both XDIFF and YDIFF are below the limit, the state will be the logic state selected in Triggering Polarity and conversely if either of them falls outside these limits, it will assume the opposite logic state. This mode can be used to signal whether or not the beam is close to the centre (beam aligned) position within a certain margin. In closed loop mode it also indicates that the controller is capable of tracking the changes in the beam position and maintain beam alignment. The lDiffThreshold parameter is specified as a percentage of full scale, in the range 1% to 99%.

0x0D TRIGOUT_SUMDIFF - This output mode is a 'logic AND' combination of the "Inside SUM range" and "Below Diff Threshold" conditions described above. Having to meet both conditions provides a more reliable indication of the normal closed loop operation when the beam is aligned and in the centre of the position sensor. In this scenario the SUM signal is within the expected limits (there is sufficient amount of light hitting the sensor) and both XDIFF and YDIFF are below a certain threshold (the beam is centralized). The second part of the condition, XDIFF and YDIFF below the threshold can also occur if the beam is blocked.

Trigger Polarity
The polarity of the trigger pulse is specified in the lTrigPolarity parameters as follows:

0x01 The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).
0x02 The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

SET

Command structure (34 bytes)

6 byte header followed by 28 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
70	08	0C	00	d	s	SubMsgID		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig1SumMin		Trig1SumMax		Trig1DiffThold		Trig2Mode		Trig2Polarity		Trig2SumMin	

24	25	26	27	28	29	30	31	32	33
Data									
Trig2SumMax		Trig2DiffThold		Reserved					

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0F,00) of the message containing the parameters	word
Trig1Mode	TRIG1 operating mode:	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) .	word
Trig1SumMin	The lower limit when the trigger mode is set to TRIGOUT_SUM	word
Trig1SumMax	The upper limit when the trigger mode is set to TRIGOUT_SUM	word
Trig1DiffThreshold	The threshold when the trigger mode is set to TRIGOUT_DIFF	word
Trig2Mode	TRIG1 operating mode	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low).	word
Trig2SumMin	The lower limit when the trigger mode is set to TRIGOUT_SUM	word
Trig2SumMax	The upper limit when the trigger mode is set to TRIGOUT_SUM	word
Trig2DiffThreshold	The threshold when the trigger mode is set to TRIGOUT_DIFF	word
Reserved		

Example: Set the Trigger parameters for KPA101 as follows:

Trig1Mode – TrigOut_SUM
Trig1Polarity – High
Trig1SumMin – 10%
Trig1SumMax – 5%
Trig1DiffThreshold – 0
Trig2Mode – Disabled
Trig2Polarity – N/A
Trig2SumMin – 0
Trig2SumMax – 0
Trig2DiffThreshold - 0

TX 70, 08, 1A, 00, D0, 01, 0F, 00, 0B, 00, 01, 00, 0A, 00, 05, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00,

Header: 70, 08, 1A, 00, D0, 01: Quad_SetParams, 30 byte data packet, Generic USB Device.
SubMsgID: 0F, 00 SetKPATrigIOConfig)
Trig1Mode – 0B, 00 TrigOut_SUM
Trig1Polarity – 01, 00 High
Trig1SumMin – 0A, 00 10%
Trig1SumMax –05, 00 5%
Trig1DiffThreshold – 0
Trig2Mode – Disabled
Trig2Polarity – N/A
Trig2SumMin – 0
Trig2SumMax – 0
Trig2DiffThreshold - 0

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	0F	00	d	s

GET:
Response structure (34 bytes):
6 byte header followed by 28 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
72	08	0C	00	d	s	SubMsgID		Trig1Mode		Trig1Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig1SumMin		Trig1SumMax		Trig1DiffThold		Trig2Mode		Trig2Polarity		Trig2SumMin	

24	25	26	27	28	29	30	31	32	33
Data									
Trig2SumMax		Trig2DiffThold		Reserved					

For structure see SET message above.

Set/Request/Get Quad_KPADigOutputs (sub-message ID = 10)

This sub-message is applicable only to KPA101 units.

Used to set the digital outputs of the KPA101 unit, if the trigger port is to be used as a general purpose digital output (i.e. trigger mode set to 0x0A TRIGOUT_GPO).
The logic state of the output can be inverted by setting the Triggering Polarity parameter to "Low"; with this option selected the state of the output will be the opposite of the corresponding bit setting in the software call. The default state of the output in this mode is also the opposite of the option selected as the Triggering Polarity.

SET
Command structure (12bytes)
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
71	08	06	00	d	s	SubMsgID		DigOPs		Reserved	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0F,00) of the message containing the parameters	word
DigOPs	The status of the digital outputs. The lowest two bits relate to TRIG1 and TRIG2	word
Reserved		

Example: Set the both Trig Outputs to ON:

TX 70, 08, 06, 00, D0, 01, 10, 00, 11, 00, 00, 00,

Header: 70, 08, 06, 00, D0, 01: Quad_SetParams, 6 byte data packet, Generic USB Device.
SubMsgID: 10, 00 SetKPATrigIOConfig)
DigOPs – 11, 00 Trig1 and Trig2 outputs set to ON (High).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
71	08	01	00	d	s

GET:
Response structure (12 bytes):
6 byte header followed by 6 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
71	08	0C	00	d	s	SubMsgID		DigOPs		Reserved	

For structure see SET message above.

MGMSG_QUAD_REQ_STATUSUPDATE 0x0880
MGMSG_ QUAD_GET_STATUSUPDATE 0x0881

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the position and status of the controller (for example position and O/P voltage). The response will be sent by the controller each time the function is requested.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
80	08	00	00	d	s

GET:
Status update messages are received with the following format:-

Response structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
81	08	0E	00	d|	s	XDiff		YDiff		Sum		XPos	

14	15	16	17	18	19
header only					
YPos		Status Bits			

Data Structure:

field	description	format
XDiff	The present X axis difference (XDIFF) signal from the detector head. (-10V to 10V in the range -32768 to 32767)	short
YDiff	The present Y axis difference (XDIFF) signal from the detector head. (-10V to 10V in the range -32768 to 32767)	short
Sum	The present Sum signal value from the detector head (0V to 10V in the range 0 to 65535)	word
XPos	The X axis position output value -10 V to 10 V (i.e. -32768 to 32767)	short
YPos	The Y axis position output value -10 V to 10 V (i.e. -32768 to 32767)	short
StatusBits	The individual bits (flags) of the 32 bit integer value are described in the following table	dword

TQD001 or TPA101 controller Status Bits

Hex Value	Bit Number	Description
0x00000001	1	Position Monitoring Mode (1 - enabled, 0 - disabled).
0x00000002	2	Open Loop Operating Mode (1 - enabled, 0 – disabled)
0x00000004	3	Closed Loop Operating Mode (1 - enabled, 0 – disabled)
0x00000008	4 to 32	For Future Use

Example

RX 81, 08, 0E, 00, 81, 50, FF, 3F, FF, 3F, FF, 7F, 00, 00, 00, 00

Header: 81, 08, 0E, 00, 81, 50: QUAD_Get_StatusUpdate, 14 byte data packet, Generic USB Device.
XDiff:.FF, 3F: 0x3FFF (16383 decimal), i.e. 5 V.
YDiff:. FF, 3F: 0x3FFF (16383 decimal), i.e. 5 V.
Sum: FF, FF: (65535 decimal), i.e. 10 V.
XPos: 00, 00 i.e. Zero
YPos: 00, 00 i.e. Zero
StatusBits: 04,00,00,00, i.e. 100 Closed Loop operating mode is enabled.

MGMSG_QUAD_ACK_STATUSUPDATE 0x0882

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called “server alive” must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
82	08	00	00	d	s

TX 82, 08, 00, 00, 21, 01

MGMSG_QUAD_SET_EEPROMPARAMS 0x0875

Function: Used to save the parameter settings for the TQD001,TPA101 or KPA101 unit. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the 'Settings' button found in the lower right hand corner of the user interface). The settings are saved for the channel specified in the Chan ID parameter

SET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
75	08	02	00	d|	s	MsgID	

Data Structure:

field	description	format
MsgID	The ID of the message parameters to be saved	word

Example:

TX 75, 08, 02, 00, D0, 01, 81, 08,

Header: 75, 08, 02, 00, D0, 01: Set_EEPROMPARAMS, 02 byte data packet, Generic USB Device.
MsgID: Save parameters specified by message 0881 (GetStatusUpdate).

TEC Control Messages

Introduction

The ActiveX functionality for the TEC Controller is accessed via the ThorlabsTEC Control Object, and provides the functionality required for a client application to control a number of T-Cube TEC Controller units.

Every hardware unit is factory programmed with a unique 8-digit serial number. This serial number is key to operation of the Thorlabs Server software and is used by the Server to enumerate and communicate independently with multiple hardware units connected on the same USB bus.

The serial number must be allocated using the HWSerialNum property, before an ActiveX control can communicate with the hardware unit. This can be done at design time or at run time.

The methods of the T-Cube TEC Controller can then be used to perform activities such as switching between display modes, reading the present TEC element temperature, and setting the LED display intensity.

For details on the use of the TEC T-Cube Controller, refer to the handbook supplied for the unit.

MGMSG_TEC_SET_PARAMS 0x0840
MGMSG_TEC_REQ_PARAMS 0x0841
MGMSG_TEC_GET_PARAMS 0x0842

Function: This generic parameter set/request message is used to control the functionality of the TEC001. The specific parameters to control are identified by the use of sub-messages. These sub messages comply with the general format of the Thorlabs message protocol but rather than having a unique first and second byte in the header carrying the "message identifier" information, the first and second byte remain the same.
Instead, for the SET and GET messages, the message identifier is carried in the first two bytes in the data packet part of the message, whilst for the REQ message it is encoded as the third byte of the header.
Likewise, when the TEC001 responds, the first two bytes of the response remain the same and the first two bytes of the data packet identify the sub-message to which the information returned in the remaining part of the data packet relates.

The following sub messages are applicable to the TEC001:

Set/Request/Get TEC_TempSetPoint (sub-message ID = 01)
Request/Get_TEC_Readings (sub-message ID = 03)
Set/Request/Get_IOSettings (sub-message ID = 05)
Request/Get_TEC_StatusBits (sub-message ID = 07)
Set/Request/Get_TEC_LoopParams (sub-message ID = 09)
Set/Request/Get TEC_Disp_Settings (sub-message ID = 0B)

To explain the principle, the following examples describe these messages in more detail.

Set/Request/Get TEC_TempSetPoint (sub-message ID = 01)
Used to set the target temperature of the TEC element associated with the ActiveX control instance.

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
40	08	04	00	d|	s	SubMsgID		TSet	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0100) of the message containing the parameters	word
TSet	Used to set the target temperature of the TEC element associated with the ActiveX control instance. Note. The units in which the temperature is returned are	word

	dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ). For a 200 kΩ. sensor the range is 0 to20000 (0 to 200 kΩ.).	

Example: Set the Temperature Setpoint for TEC001 as follows:
TSet: 65 °C

TX 40, 08, 04, 00, D0, 01, 01, 00, 64, 19

Header: 70, 08, 08, 00, D0, 01: TEC_SetTempSetPoint, 4 byte data packet, Generic USB Device.
SubMsgID: 01, 00 SetTempSetPoint
TSet: *64, 19 ,(6500):* Set the set point to 65 °C

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	01	00	d	s

GET:
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
42	08	04	00	d|	s	SubMsgID		TSet	

For structure see Set message above.

Request/Get TEC_Readings (sub-message ID = 3)
This message returns the present readings of the TEC unit as follows:
ITec The TEC output current in mA. (0 to 2000mA in the range -0 to 2000)
TAct The actual temperature of the TEC element associated with the ActiveX control instance.
Note. The units in which the temperature is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ For a 200 kΩ sensor the range is 0 to20000 (0 to 200 kΩ).
TSet The temperature setpoint of the TEC element associated with the ActiveX control instance.
Note. The units in which the setpoint is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ For a 200 kΩ sensor the range is 0 to20000 (0 to 200 kΩ).

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	03	00	d	s

TX 41, 08, 03, 00, 50, 01,

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
42	08	08	00	d|	s	SubMsgID		ITec		TAct		TSet	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0300) of the message containing the parameters	word
ITec	Returns the TEC output current in mA. (0 to 2000mA in the range -0 to 2000)	short
TAct	Returns the present temperature of the TEC element associated with the ActiveX control instance. Note. The units in which the temperature is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ). For a 200 kΩ. sensor the range is 0 to20000 (0 to 200 kΩ.).	short

TSet	Returns the target temperature of the TEC element associated with the ActiveX control instance. Note. The units in which the temperature is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ). For a 200 kΩ. sensor the range is 0 to20000 (0 to 200 kΩ.).	word

Example: Get the Quad Detector T-Cube readings (T-Cube in open loop mode)

RX 42, 08, 08, 00, D0, 01, 03, 00, E8, 03, DC, 05, 40, 1F,

Header: 42, 08, 08, 00, D0, 01: TEC_GetPARAMS, 8 byte data packet, Generic USB Device.
MsgID: 03, 00: Get Quad Readings
ITec:.E8, 03: 0x03E8 (1000 decimal), i.e. 1 V.
TAct:. DC, 05: 0x05DC (1500 decimal), i.e. 1.5 V.
TSet: 40, 1F: 0x1F40 (8000 decimal), i.e. 80 °C.

Set/Request/Get IOSettings (sub-message ID = 5)

This message sets the type of TEC element associated with the ActiveX control instance.
If an AD59x transducer is selected, the temperature is set and displayed in °C.
If a 20kOhm or 200kOhm thermistor is selected, the temperature is set and displayed in kOhms.

SET:

Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
40	08	06	00	d|	s	SubMsgID		wSensor		sILim	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0500) of the message containing the parameters	word
wSensor	This parameter contains constants that specify the type of TEC element controlled by the unit. 0 SENSOR_IC_AD59X TEC element is a AD59x IC type transducer. 1 SENSOR_THERM20KOHM TEC element is a 20kOhm thermistor. 2 SENSOR_THERM200KOHM TEC element is a 200kOhm thermistor.	word
sILim	This parameter returns the maximum current that the TEC controller associated with the ActiveX control instance can source into the TEC element. Values are set in the range 0 to 2000 (0 to 2000 mA).	short

Example: Set the TEC IO Settings as follows

RX 40, 08, 0C, 00, D0, 01, 05, 00, 01, 00, 01, 80

Header: 42, 08, 0C, 00, D0, 01: TEC_SetPARAMS, 6 byte data packet, Generic USB Device.
SubMsgID: 05, 00: Set TEC_IOSettings
wSensor:.01, 00: 0x0001 i.e. AD59x IC type transducer.
sILim:. E8, 03: 0x03E8 (10000 decimal), i.e. 1A.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	05	00	d	s

TX 41, 08, 05, 00, 50, 01,

GET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
42	08	06	00	d|	s	SubMsgID		wSensor		sILim	

See Set message for structure

Request/Get TEC_Status Bits (sub-message ID = 7)
This sub command can be used to request the TEC001 status bits. The message only has a request/ get part.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	07	00	d	s

TX 41, 08, 07, 00, 50, 01,

GET:
Status update messages are received with the following format:-

Response structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
42	08	06	00	d|	s	SubMsgID		StatusBits			

Data Structure:

field	description	format
MsgID	The message ID (0700) of the message containing the parameters	word
StatusBits	The individual bits (flags) of the 32 bit integer value are described in the following table.	dword

TEC controller Status Bits

Hex Value	Bit Number	Description
0x00000001	1	TEC output enabled state (1 - enabled, 0 - disabled).
	2 to 4	For Future Use
0x00000010	5	Display mode (1 – TAct, 0 - else).
0x00000020	6	Display mode (1 – TSet, 0 - else).
0x00000040	7	Display mode (1 – TDelta, 0 - else).
0x00000080	8	Display mode (1 – ITec, 0 - else).
	9 to 30	For Future Use
0x40000000	31	Error
0x80000000	32	For Future Use

Example
RX 42, 08, 06, 00, 81, 50, E8, 03, DC, 05, 40, 1F, 11, 00, 00, 00

Header: 42, 08, 06, 00, 81, 50: TEC_SetParams, 6 byte data packet, Generic USB Device.
SubMsgID: 07, 00: Set TEC_StatusBits

StatusBits: 11,00,00,00, 0X00000011 (17 decimal) i.e. TEC is enabled with Tact display mode selected. No errors.

Set/Request/Get TEC_LoopParams (sub-message ID = 9)

Used to set the proportional, integration and derivative feedback loop constants to the value specified in the PGain, IGain and DGain parameters respectively. They apply when the TEC unit is operated in closed loop mode, and demand signals are generated at the rear panel connectors by the feedback loops. These demand signals act to drive the heating element to the temperature required.

When operating in closed loop mode, the proportional, integral and derivative (PID) constants can be used to fine tune the behaviour of the dual feedback loops to adjust the response of the temperature demand output current. The feedback loop parameters need to be adjusted to suit the different types of sensor that can be connected to the system.

SET:

Command structure (14 bytes)

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
70	08	08	00	d|	s	SubMsgID		PGain		IGain		DGain	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 09,00) of the message containing the parameters	word
PGain	The proportional gain. This term provides the force used to drive the output to the demand set point, reducing the positional error. Together with the Integral and Derivative, these terms determine the system response characteristics and accept values in the range 1 to 100 (i.e. 1 to 100 in Thorlabs User GUI).	word
IGain	The integral gain. This term provides the ‘restoring’ force that grows with time, ensuring that the set point error is eventually reduced to zero. Together with the Proportional and Derivative, these terms determine the system response characteristics and accept values in the range 0 to 100 (i.e. 0 to 100 in Thorlabs User GUI).	word
DGain	The derivative gain. This term provides the ‘damping’ force proportional to the rate of change of the temperature. Together with the Proportional and Integral, these terms determine the system response characteristics and accept values in the range 0 to 100 (i.e. 0 to 100 in Thorlabs User GUI).	word

Example: Set the PID parameters for TEC001 as follows:
Proportional: 65
Integral: 80
Derivative: 60

TX 40, 08, 08, 00, D0, 01, 09, 00, 41, 00, 50, 00, 3C, 00,

Header: 40, 08, 08, 00, D0, 01: TEC_SetParams, 8 byte data packet, Generic USB Device.

SubMsgID: 09, 00 Set_TECLoopParams)
PGain: *41, 00:* Set the proportional term to 65
IGain: *50, 00: (32767x80/100)*: Set the integral term to 80
DGain: 3C, 00: *(32767x60/100)*: Set the derivative term to 60

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	09	00	d	s

GET:
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
72	08	08	00	d|	s	SubMsgID		PGain		IGain		DGain	

For structure see Set message above.

Set/Request/Get TEC Display Settings (sub-message ID = 0B)

This message can be used to adjust or read the front panel LED display brightness and the display units.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
40	08	08	00	d|	s	SubMsgID		DispIntensity		DispMode		Unused	

Data Structure:

field	description	format
MsgID	The message ID (i.e. 0B00) of the message containing the parameters	word
DispIntensity	The intensity is set as a value from 0 (Off) to 255 (brightest).	word
DispMode	The LED display window on the front of the unit can be set to display four different values; the actual temperature of the TEC element (TAct), the difference between the actual temperature and the set point (TDelta), the applied current (ITec), or the demanded set point value (TSet). 0 DISPMODE_TACT the display shows the actual temperature of the TEC element 1 DISPMODE_TSET the display shows the demanded set point value. 2 DISPMODE_DELTA the display shows the difference between the actual temperature (TAct) and the set point temperature (TSet). 3 DISPMODE_ITEC the display shows the current (in Amps) sourced into the TEC element by the controller.	word
Reserved	N/A	word

Example: Set the display to max brightness and the display mode to TAct

TX 40, 08, 08, 00, D0, 01, 0B, 00, FF, 00, 01, 00, 00, 00

Header: 40, 08, 08, 00, D0, 01: TEC_SetParams, 08 byte data packet, Generic USB Device.
SubMsgID: *0B, 00: Set Display Settings*
DispIntensity: *FF, 00*: Sets the display brightness to 255 (100%)
DispMode: 01, 00 Sets the display to show the actual temperature of the TEC element.

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
41	08	0B	00	d	s

Example: TX 41, 08, 0B, 00, 50, 01

GET:

Command structure (14 bytes)

6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
42	08	08	00	d|	s	SubMsgID		DispIntensity		DispMode		Unused	

See SET for data structure.

MGMSG_TEC_SET_EEPROMPARAMS 0x0850

Function: Used to save the parameter settings for the TEC001 unit. These settings may have been altered either through the various method calls or through user interaction with the GUI (specifically, by clicking on the 'Settings' button found in the lower right hand corner of the user interface).

SET:
Command structure (8 bytes)
6 byte header followed by 2 byte data packet as follows:

0	1	2	3	4	5	6	7
header						*Data*	
50	08	02	00	d|	s	SubMsgID	

Data Structure:

field	description	format
SubMsgID	For future use	word

Example:

TX 75, 08, 02, 00, D0, 01, 00, 00,

Header: E7, 07, 04, 00, D0, 01: Set_EEPROMPARAMS, 02 byte data packet, Generic USB Device.

MGMSG_TEC_REQ_STATUSUPDATE 0x0860
MGMSG_ TEC_GET_STATUSUPDATE 0x0861

Function: This function is used in applications where spontaneous status messages (i.e. messages sent using the START_STATUSUPDATES command) must be avoided.
Status update messages contain information about the output current and actual temperature of the transducer. The response will be sent by the controller each time the function is requested.

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
60	08	00	00	d	s

GET:

Status update messages are received with the following format:-

Response structure (16 bytes)

6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
61	08	0E	00	d|	s	ITec		TAct		TSet	

12	13	14	15
header only			
Status Bits			

Data Structure:

field	description	format
ITec	The TEC output current in mA. (0 to 2000mA in the range -0 to 2000)	short
TAct	The actual temperature of the TEC element associated with the ActiveX control instance. Note. The units in which the temperature is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°). For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ). For a 200 kΩ. sensor the range is 0 to20000 (0 to 200 k Ω.).	short
TSet	The temperature setpoint of the TEC element associated with the ActiveX control instance. Note. The units in which the setpoint is returned are dependent upon the 'Sensor Type' selected (via the Settings panel or by calling the SetTempSetPoint submessage). If an IC type sensor is selected, the set point temperature is displayed in °C in the range -4500 to 14500 (45.0° to 145.0°).	word

	For a 20 kΩ.thermistor sensor, the set point is displayed in kΩ in the range 0 to 2000 (0 to 20 kΩ). For a 200 kΩ. sensor the range is 0 to20000 (0 to 200 k Ω.).	
StatusBits	The individual bits (flags) of the 32 bit integer value are described in the following table	dword

TEC controller Status Bits

Hex Value	**Bit Number**	**Description**
0x00000001	1	TEC output enabled state (1 - enabled, 0 - disabled).
	2 to 4	For Future Use
0x00000010	5	Display mode (1 – TAct, 0 - else).
0x00000020	6	Display mode (1 – TSet, 0 - else).
0x00000040	7	Display mode (1 – TDelta, 0 - else).
0x00000080	8	Display mode (1 – ITec, 0 - else).
	9 to 30	For Future Use
0x40000000	31	Error
0x80000000	32	For Future Use

Example

RX 61, 08, 0A, 00, 81, 50, E8, 03, DC, 05, 40, 1F, 11, 00, 00, 00

Header: 61, 08, 0A, 00, 81, 50: TEC_Get_StatusUpdate, 10 byte data packet, Generic USB Device.
ITec:.E8, 03: 0x03E8 (1000 decimal), i.e. 1 V.
TAct:. DC, 05: 0x05DC (1500 decimal), i.e. 1.5 V.
TSet: 40, 1F: 0x1F40 (8000 decimal), i.e. 80 °C.
StatusBits: 11,00,00,00, 0X00000011 (17 decimal) i.e. TEC is enabled with Tact display mode selected. No errors.

MGMSG_TEC_ACK_STATUSUPDATE 0x0862

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called "server alive" must be sent by the server to the controller at least once a second or the controller will stop responding after ~50 commands.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and it if has sent 50 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5

header only					
82	08	00	00	d	s

TX 62, 08, 00, 00, 21, 01

TIM and KIM Control Messages

Introduction

The functionality for the TIM101 and KIM101 Piezo Motor Controllers is accessed via the ThorlabsPZMOT Control Object, and provides the functionality required for a client application to control a number of Controller units.
Every hardware unit is factory programmed with a unique 8-digit serial number. This serial number is key to operation of the Thorlabs Server software and is used by the Server to enumerate and communicate independently with multiple hardware units connected on the same USB bus.
The serial number must be allocated using the HWSerialNum property, before an ActiveX control can communicate with the hardware unit. This can be done at design time or at run time.
The methods of the Piezo Motor Controller can then be used to perform activities such as setting the drive voltage, setting the jog step size and setting top panel control parameters.

Note. The channel being addressed must be enabled by calling the Set_ChanEnableState method, before the following methods can be used.

For details on the use of the TIM101 and KIM101 Controller units, refer to the handbook available to download from www.thorlabs.com.

MGMSG_PZMOT_SET_PARAMS 0x08C0
MGMSG_ PZMOT_REQ_PARAMS 0x08C1
MGMSG_ PZMOT_GET_PARAMS 0x08C2

Function: This generic parameter set/request message is used to control the functionality of the TIM101 and KIM101 controllers. The specific parameters to control are identified by the use of sub-messages. These sub messages comply with the general format of the Thorlabs message protocol but rather than having a unique first and second byte in the header carrying the "message identifier" information, the first and second byte remain the same.
Instead, for the SET, REQ and GET messages, the message identifier is carried in the first two bytes in the data packet (7 and 8) part of the message,
Likewise, when the unit responds, the first two bytes of the response remain the same and the first two bytes of the data packet identify the sub-message to which the information returned in the remaining part of the data packet relates.

The following sub messages are applicable to the TIM101:

Set/Request/Get_PZMOT_PosCounts (sub-message ID = 05)
Set/Request/Get_PZMOT_DriveOPParams (sub-message ID = 07)
Set/Request/Get_TIM_JogParameters (sub-message ID = 09)
Set/Request/Get TIM_PotParameters (sub-message ID = 11)
Set/Request/Get TIM_ButtonParameters (sub-message ID = 13)

The following sub-messages are applicable to the KIM101:

Set/Request/Get_PZMOT_PosCounts (sub-message ID = 05)
Set/Request/Get_PZMOT_DriveOPParams (sub-message ID = 07)
Set/Request/Get_PZMOT_LimitSwitchParams (sub-message ID = 0B)
Request/Get_PZMOT_HomeParams (sub-message ID = 0F)
Set/Request/Get_PZMOT_KCubeMMIParams (sub-message ID = 15)
Set/Request/Get_PZMOT_TrigIOConfig (sub-message ID = 17)
Set/Request/Get_PZMOT_TrigParams (sub-message ID = 19)
Set/Request/Get_PZMOT_ChanEnableMode (sub-message ID = 2B)
Set/Request/Get_PZMOT_KCubeJogParams (sub-message ID = 2D)
Set/Request/Get_PZMOT_KCubeFeedbackSigParams (sub-message ID = 30)
Set/Request/Get_PZMOT_KCubeMoveRelativeParams (sub-message ID = 32)
Set/Request/Get_PZMOT_KCubeMoveAbsoluteParams (sub-message ID = 34)

The following sub-messages are applicable to PDXC2:
Set/Request/Get_PZMOT_ CloseLoopParams (sub-message ID = 39)
Set/Request/Get_PZMOT_EtmConfig (sub-message ID = 3B)
Set/Request/Get_PZMOT_EtmParams (sub-message ID = 3C)
Request/Get_PZMOT_TPOS (sub-message ID = 3D)
Set/Request/Get_PZMOT_EthernetParams (sub-message ID = 3F)
Request/Get_PZMOT_CURRENT_POS (sub-message ID = 40)
Set/Request/Get_PZMOT_AMD (sub-message ID = 43)
Set/Request/Get_PZMOT_JogParams (sub-message ID = 44)

Set/Request/Get_PZMOT_AmpOutParams (sub-message ID = 45)
Set/Request/Get_PZMOT_OpenMoveParams (sub-message ID = 46)
Set/Request/Get_PZMOT_CloseMoveParams (sub-message ID = 47)

The examples on the following pages describe these messages in more detail.

Set/Request/Get_PZMOT_PosCounts (sub-message ID = 05)
Applicable to both TIM101 and KIM101

This sub-message sets/returns the position counter value and is used to set the counter to zero when the motor is at the required zero position. All absolute moves are then measured from this zeroed position.

SET:
Command structure (18 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
C0	08	0E	00	d|	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17
Data							
Position				EncCount			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0500) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
Position	The position counter value, specified in number of steps must be set to zero.	long
EncCount	Not Used	long

Example: Set the TIM Position Counter

TX C0, 08, 0C, 00, D0, 01, 05, 00, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00

Header: C0, 08, OC, 00, D0, 01: PZMOT_SET_PARAMS, 12 byte data packet, USB Device.
SubMsgID: 05, 00 *Set_TIM_PositionCounters*
ChanIdent: 01, 00 Channel 1
Position: 00, 00, 00, 00 Zero
EncCount: 00, 00, 00, 00 Not Used

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5

header only					
C1	08	05	01	d	s

TX C1, 08, 05, 01, D0, 01,

GET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
C2	08	0E	00	d|	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17
Data							
Position				EncCount			

See Set message for structure

Set/Request/Get_DriveOPParameters (sub-message ID = 07)
Applicable to both TIM101 and KIM101

This sub-message sets various drive parameters which define the speed and acceleration of moves initiated in the following ways:

- by clicking in the position display
- via the top panel controls when 'Go To Position' mode is selected (in the Set_TIM_JogParameters (09) or Set_KCubeMMIParams (15) sub-messages).
- via software using the MoveVelocity, MoveAbsoluteStepsEx or MoveRelativeStepsEx methods.

Note. Drive parameters for Jog moves are specified in the Set_TIM_JogParameters sub-message.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	0E	00	d|	s	SubMsgID		ChanIdent		MaxVoltage	

12	13	14	15	16	17	18	19
Data							
StepRate				StepAccn			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0700) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
MaxVoltage	The maximum piezo drive voltage, in the range 85V to 125V.	word
StepRate	The piezo motor moves by ramping up the drive voltage to the value set in the MaxVoltage parameter and then dropping quickly to zero, then repeating. One cycle is termed a step. This parameter specifies the velocity to move when a command is initiated. The step rate is specified in steps/sec, in the range 1 to 2,000.	long
StepAccn	This parameter specifies the acceleration up to the step rate, in the range 1 to 100,000 cycles/sec/sec.	long

Example: Set the TIM Drive Params

TX C0,08,0E,00,81,50,07,00,01,00,6E,00,F4,01,00,00,A0,86,01,00

Header: C0, 08, OE, 00, 81, 50: PZMOT_SET_PARAMS, 18 byte data packet, USB Device.
SubMsgID: 07, 00 *Set_TIM_DriveParameters*

ChanIdent: 01, 00	Channel 1	
MaxVoltage: 6E, 00	100V	(6E)
StepRate: F4, 01, 00, 00	500 Steps/Sec	(01F4)
StepAccn: A0, 86, 01, 00	10,000 Steps/Sec/Sec	(0186A0

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	07	01	d	s

TX C1, 08, 07, 01, 50, 01,

GET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	0E	00	d|	s	SubMsgID		ChanIdent		MaxVoltage	

12	13	14	15	16	17	18	19
Data							
StepRate				StepAccn			

See Set message for structure

Set/Request/Get_TIM_JogParameters (sub-message ID = 09)
Applicable only to TIM101 units

This sub-message sets various jog parameters which define the speed and acceleration of moves initiated in the following ways:

by clicking the jog buttons on the GUI panel
by pressing the buttons on the unit when 'Single Step' mode is selected.
via software using the MoveJog method.

Note. Drive parameters for motor moves are specified in the Set_TIM_DriveParameters sub-message.

SET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	12	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JogStepSize				JogStepRate				JogStepAccn			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0900) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
JogMode	Jog commands can be issued by calling the MoveJog method, via the Motor Control GUI panel or by pressing the buttons on the hardware unit. When a jog command is received, if the jog mode is set to 1 (i.e. 'Continuous') the motor continues to move until the jog signal is removed (i.e. the jog button is released) when the motor will stop immediately. If the mode is set to '2' (i.e. Single Step) the motor moves by the step size specified in the JogStepSize parameter.	word
JogStepSize	A jog step consists of a number of drive pulses. This parameter specifies the number of pulses which make up a jog step, in the range 1 to 2,000.	long
JogStepRate	The piezo motor moves by ramping up the drive voltage to the value set in the Set_TIM_DriveParameters sub-message and then dropping quickly to zero, then repeating. One cycle is termed a step. This parameter specifies the velocity to move when a command is initiated. The step rate is specified in steps/sec, in the range 1 to 2,000	long
JogStepAccn	This parameter specifies the acceleration up to the step rate, in the range 1 to 100,000 cycles/sec/sec.	long

Example: Set the TIM Jog Parameters

TX C0,08,12,00,81,50,09,00,01,00,02,00,FA,00,00,00,F4,01,00,00,A0,86,01,00

Header: C0, 08, 12, 00, 81, 50: PZMOT_SET_PARAMS, 18 byte data packet, Generic USB Device.

SubMsgID: 09, 00	*Set_TIM_JogParameters*	
ChanIdent: 01, 00	Channel 1	
JogMode: 02, 00	Single Step Jog Mode	
JogStepSize: FA. 00, 00, 00	250 steps (FA)	
JogStepRate: F4, 01, 00, 00	500 Steps/Sec	(01F4)
JogStepAccn: A0, 86, 01, 00	10,000 Steps/Sec/Sec	(0186A0

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	09	01	d	s

TX C1, 08, 09, 01, 50, 01,

GET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	12	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JogStepSize				JogStepRate				JogStepAccn			

See Set message for structure

Set/Request/Get_TIM_PotParameters (sub-message ID = 11)
Applicable only to TIM101 units

This sub-message defines the speed of a move initiated by the potentiometer on the top panel of the hardware unit.
The potentiometer slider is sprung such that when released it returns to its central position. In this central position the piezo motor is stationary. As the slider is moved away from the centre, the motor begins to move. Bidirectional control of the motor is possible by moving the slider in both directions. The speed of the motor increases as a function of slider deflection.

SET:
Command structure (14 bytes). 6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	08	00	d|	s	SubMsgID		ChanIdent		MaxStepRate			

Data Structure:

field	description	format
MsgID	The message ID (i.e. 11,00) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
MaxStepRate	The speed (in drive pulses per second) of a move initiated by the top panel potentiometer, in the range 1 to 2,000.	long

Example: Set the TIM Pot Parameters

TX C0,08,08,00,81,50,11,00,01,00,E8,03,00,00

Header: C0, 08, 08, 00, 81, 50: TIM_SetParams, 08 byte data packet, Generic USB Device.
SubMsgID:11, 00: *Set_TIM_PotParams*
ChanIdent: 01, 00 Channel 1
MaxStepRate: E8, 03, 00, 00 1000 (03E8) pulses per second

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	09	01	d	s

TX C1, 08, 11, 01, 50, 01,

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C2	08	08	00	d|	s	SubMsgID		ChanIdent		MaxStepRate			

See SET for data structure.

Set/Request/Get_TIM_ButtonParameters (sub-message ID = 13)
Applicable only to TIM101 units

The buttons on the top of the unit can be used either to jog the motor, or to perform moves to absolute positions. This sub-message sets the operation mode of the buttons.

SET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	12	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Position1				Position2				TimeOut1		TimeOut2	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 1300) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
Mode	This parameter specifies the mode of operation of the buttons. If set to '1' (Jog Mode), the front panel buttons are used to jog the motor. Once set to this mode, the move parameters for the buttons are taken from the 'Jog' parameters set via the 'Set_TIM_JogParameters sub-message. If set to '2' (Position Mode) each button can be programmed with a different position value (as set in the Position1 and position2 parameters below), such that the controller will move the motor to that position when the specific button is pressed.	word
Position1	This parameter is applicable only if Position mode is selected above, and is the position to which the motor will move when the top button is pressed. The position is set in number of steps, measured from the zero position.	long
Position2	This parameter is applicable only if Position mode is selected above, and is the position to which the motor will move when the bottom button is pressed. The position is set in number of steps, measured from the zero position.	long
TimeOut1	For Future Use	word
TimeOut2	For Future Use	word

Example: Set the TIM Button Parameters

TX C0,08,12,00,81,50,13,00,01,00,01,00,C8,00,00,00,F4,01,00,00,FA,00,FA,00

Header: C0, 08, 12, 00, 81, 50: PZMOT_SET_PARAMS, 18 byte data packet, Generic USB Device.

SubMsgID: 13, 00	*Set_TIM_ButtonParameters*
ChanIdent: 01, 00	Channel 1
Mode: 01, 00	Jog Mode
Position1: C8. 00, 00, 00	200 steps from the zero position
Position2: F4, 01, 00, 00	500 steps from the zero position
TimeOut1: FA, 00,	Not Used
TimeOut2: FA, 00,	Not Used

REQUEST:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	13	01	d	s

TX C1, 08, 13, 01, 50, 01,

GET:

Command structure (24 bytes)

6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	12	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Position1				Position2				TimeOut1		TimeOut2	

See Set message for structure

Set/Request/Get_PZMOT_LimSwitchParams (sub-message ID = 0B)

This message is not implemented at this time and is for future use with encoder-equipped actuators. Applicable only to KIM001 and KIM101 units

The action that the forward and reverse hardware limit switches make on contact is inherent in the design of the stage being driven. This sub-message notifies the system to the action of the limit switches associated with the stage/actuator being driven by the channel specified.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	0A	00	d|	s	SubMsgID		ChanIdent		FwdHardLimit	

12	13	14	15
Data			
RevHardLimit		StageID	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0B00) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
FwdHardLimit	The operation of the Forward hardware limit switch when contact is made. 0x01 Ignore switch or switch not present. 0x02 Switch makes on contact. 0x03 Switch breaks on contact. 0x04 Switch makes on contact - only used for homes (e.g. limit switched rotation stages). 0x05 Switch breaks on contact - only used for homes (e.g. limit switched rotations stages).	word
RevHardLimit	The operation of the Reverse hardware limit switch when contact is made – see FWDHardLimit for parameter values.	word
StageID	Not Used	word

Example: Set the KIM Limit Switch Parameters

TX C0,08,0A,00,81,50, 0B,00,01,00,02,00,02,00,00,00,

Header: C0, 08, 12, 00, 81, 50: PZMOT_SET_PARAMS, 10 byte data packet, Generic USB Device.
SubMsgID: 0B, 00 *Set_LimSwitchParams*
ChanIdent: 01, 00 Channel 1
FwdHardLimit: 02, 00 Switch makes on contact
RevHardLimit: 02, 00 Switch makes on contact

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	0B	01	d	s

TX C1, 08, 13, 01, 50, 01,

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	0A	00	d|	s	SubMsgID		ChanIdent		FwdHardLimit	

12	13	14	15
Data			
RevHardLimit		StageID	

See Set message for structure

Request/Get_PZMOT_HomeParams (sub-message ID = 0F)
Applicable only to KIM001 and KIM101 units
Note. This message is for future use with closed loop homing applications and is not yet implemented. It is shown for reference only.

Used to set the home parameters for the stage/actuator associated with the specified motor channel.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	0F	00	d	s

GET:
Command structure (22 bytes)
6 byte header followed by 16 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	10	00	d|	s	SubMsgID		ChanIdent		HomeDirection	

12	13	14	15	16	17	18	19	20	21
Data									
HomeLimSwitch		HomeStepRate				HomeOffsetDist			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 0F00) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
HomeDirection	The direction sense for a move to Home, either 1 - Forward/Positive or 2 - Reverse/negative.	word
HomeLimSwitch	The limit switch associated with the home position 1 - Forward or 2 - Reverse	word
HomeStepRate	The homing velocity (i.e. step rate) in position steps/sec. A 4 byte unsigned long value.	long
HomeOffsetDist	The distance of the Home position from the Home Limit Switch. This is a 4 byte signed integer that specifies the offset distance in position steps, in the range 0 to 10000.	long

Example: Set the home parameters for chan 2 as follows:
Home Direction: Reverse.
Limit Switch: Reverse
Home Vel: 1000 steps/sec
Offset Dist: 500 steps.

TX C2, 08, 10, 00, 81, 50, 0F, 00, 02, 00, 02, 00, 02, 00, E8. 03, 00, 00, F4, 01, 00, 00,

Header: C2, 08, 10, 00, A2, 01: Get KIM HomeParams, 16 byte data packet, Generic USB Device
SubMsg ID: 0F, 00
Chan Ident: 02, 00: Channel 2
HomeDirection: *02, 00:* Reverse
HomeLimSwitch: 02, 00: Reverse
HomeStepRate: E8, 03, 00, 00: 1000 steps/sec
Offset Distance: F4, 01, 00, 00: 500 Step Offset

Set/Request/Get_PZMOT_KCubeMMIParams (sub-message ID = 15)
Applicable only to KIM001 and KIM101 units

This sub-message is used to configure the operating parameters of the top panel Joystick.

SET
Command structure (30 bytes)
6 byte header followed by 24 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	1C	00	d	s	SubMsg ID		ChanIdent		JSMode	

12	13	14	15	16	17	18	19	20	21
Data									
JSMaxStepRate				JSDirSense		PreSetPos1			

22	23	24	25	26	27	28	29
Data							
PreSetPos2				DispBrightness		Reserved	

Data Structure:

field	description	format
SubMsg ID	The message ID (i.e. 1500) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
JSMode	This parameter specifies the operating mode of the joy stick as follows: 1 Velocity Control Mode - Deflecting the joystick starts a move with the velocity proportional to the deflection. The maximum velocity (i.e. velocity corresponding to the full deflection of the joystick) is specified in the JSMaxStepRate and parameter following. 2 Jog Mode - Deflecting the joystick initiates a jog move, using the parameters specified by the PZMOT_JogParams sub-message. Keeping the joystick deflected repeats the move automatically after the current move has completed. 3 Go To Position Mode - Deflecting the joystick starts a move from the current position to one of the two predefined “teach” positions. The teach positions are specified in number of steps from the home position in the PresetPos1 and PresetPos2 parameters. For the KIM101 unit, move the joystick left (Ch1 and 3) or up (Ch 2 and 4) to go to position 1, and right or down to go to position 2. For the KIM001 unit, move the joystick up to go to position 1, and down to go to position 2.	word
JSMaxStepRate	The max velocity of a move initiated by the top panel joystick (i.e. the max step rate for full joystick deflection), in the range 1 to 2000 position steps/sec.	long

JSDirSense	This parameter specifies the direction of a move initiated by the joystick as follows: 0 Joystick initiated moves are disabled. The joystick is used for menuing only. 1 Upwards/Right deflection of the joystick results in a positive motion (i.e. increased position count). The following option applies only when the JSMode is set to Velocity Control Mode (1). If set to Jog Mode (2) or Go to Position Mode (3), the following option is ignored. 2 Upwards/Right deflection of the joystick results in a negative motion (i.e. decreased position count).	word
PresetPos1	The preset position 1 when operating in go to position mode, measured in position steps from the home position.	long
PresetPos2	The preset position 2 when operating in go to position mode, measured in position steps from the home position.	long
DispBrightness	In certain applications, it may be necessary to adjust the brightness of the LCD display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again.	word

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	15	00	d	s

Example: Request the settings for the top panel joystick

TX C1, 08, 15, 00, 50, 01

GET:
Response structure (6 bytes):

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	1C	00	d	s	SubMsg ID		ChanIdent		JSMode	

12	13	14	15	16	17	18	19	20	21
Data									
JSMaxStepRate				JSDirSense		PreSetPos1			

22	23	24	25	26	27	28	29
Data							
PreSetPos2				DispBrightness		Reserved	

For structure see SET message above.

Set/Request/Get_PZMOT_KCubeTrigIOConfig (sub-message ID = 17)
Applicable only to KIM001 and KIM101 units

The KIM101 K-Cube inertial piezo motor controller has two bidirectional trigger ports (I/O 1 and I/O 2) that can be used as a general purpose digital input/output, or can be configured to output a logic level to control external equipment.
When the port is used as an output it provides a push-pull drive of 5 Volts, with the maximum current limited to approximately 8 mA. The current limit prevents damage when the output is accidentally shorted to ground or driven to the opposite logic state by external circuitry. The active logic state can be selected High or Low to suit the requirements of the application.

This sub-message sets the operating parameters of the I/O 1 and I/O 2 connectors on the front panel of the unit.

Warning. Do not drive the TRIG ports from any voltage source that can produce an output in excess of the normal 0 to 5 Volt logic level range. In any case the voltage at the TRIG ports must be limited to -0.25 to +5.25 Volts.

Trigger Modes
Input Trigger Modes
When configured as an input, the TRIG ports can be used as a general purpose digital input, or for triggering a drive voltage change as follows:

0x00 DISABLED - The trigger IO is disabled.
0x01 GPI - General purpose logic input (read through status bits using the PZ_GET_PZSTATUSUPDATE message).
0x02 RELMOVE - Input trigger for a relative move. On receipt of the trigger, the motor will move by the number of position steps entered in the PZMOT_KCubeMoveRelativeParams sub-message (0x32).
0x03 ABSMOVE - Input trigger for an absolute move. On receipt of the trigger, the motor will move to the absolute position entered in the PZMOT_KCubeMoveAbsoluteParams sub-message (0x34).
0x04 RESETCOUNT - Input trigger for count reset. On receipt of the trigger, the counter will reset and all subsequent moves will be measured from the current position.

When used for triggering a move, the port is edge sensitive. In other words, it has to see a transition from the inactive to the active logic state (Low->High or High->Low) for the trigger input to be recognized. For the same reason a sustained logic level will not trigger repeated moves. The trigger input has to return to its inactive state first in order to start the next trigger.

Output Trigger Modes
When configured as an output, the TRIG ports can be used as a general purpose digital output.

0x0A GPO - General purpose logic output (set using the MOD_SET_DIGOUTPUTS message).
0x0B INMOTION - Trigger output active (level) when motor 'in motion'. The output trigger goes high (5V) or low (0V) (as set in the lTrig1Polarity and lTrig2Polarity parameters) when the stage is in motion.

0x0C MAXVELOCITY - Trigger output active (level) when motor is at 'max velocity'. The max velocity limit that generates the trigger is dependent on the type of move being performed, e.g. jog move, joystick move etc.
0x10 FWDLIMIT - Trigger output active (level) when the FWD limit switch is activated.
0x11 REVLIMIT - Trigger output active (level) when the REV limit switch is activated.
0x12 EITHERLIMIT - Trigger output active (level) when the either the FWD or REV limit switch is activated.

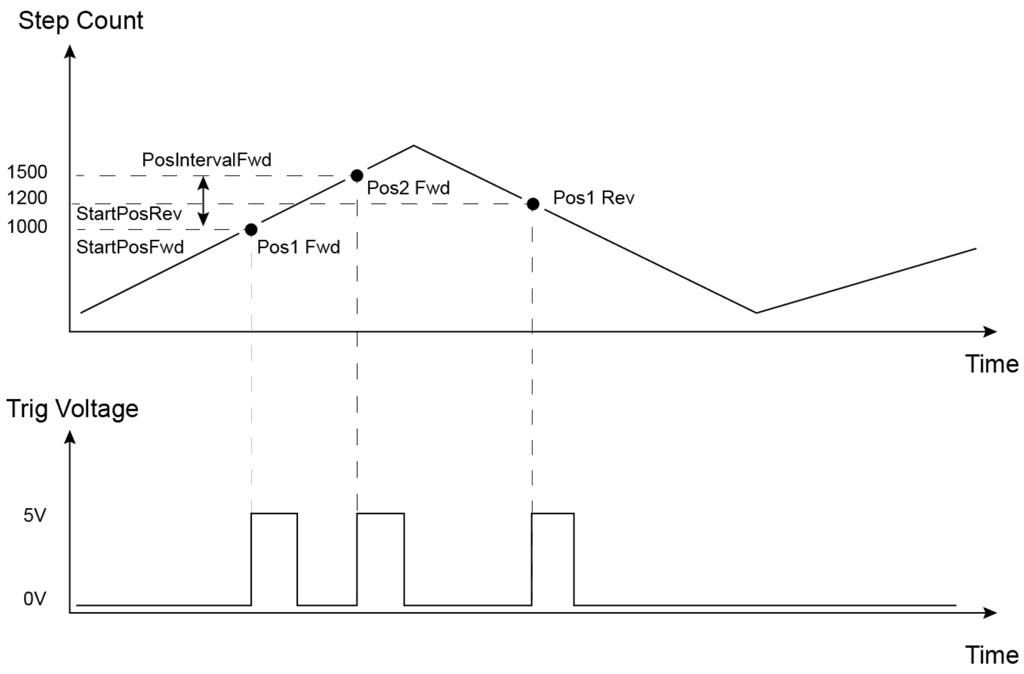
The following modes can be set to only one trigger at a time.

0x0D POSSTEPS_FWD - Trigger output active (pulsed) at pre-defined positions moving forward (set using StartPosFwd, IntervalFwd, NumPulsesFwd and PulseWidth parameters in the SetKCubeTrigParams message) – see Trigger Out Position Steps section below. Only one Trigger port at a time can be set to this mode.
0x0E POSSTEPS_REV - Trigger output active (pulsed) at pre-defined positions moving backwards (set using StartPosRev, IntervalRev, NumPulsesRev and PulseWidth parameters in the SetKCubeTrigParams message) – see Trigger Out Position Steps section below. Only one Trigger port at a time can be set to this mode.
0x0F POSSTEPS_BOTH Trigger output active (pulsed) at pre-defined positions moving forwards and backward – see Trigger Out Position Steps section below. Only one Trigger port at a time can be set to this mode.

Trigger Out Position Steps

In the three position step modes described above, the controller outputs a configurable number of pulses, of configurable width, when the actual position of the stage matches the position values configured as the Start Position and Position Interval - see SetKCubeTrigParams message. These modes allow external equipment to be triggered at exact position values (measured in number of steps).

Using the POSSTEPS modes above, position triggering can be configured to be unidirectional (forward or reverse only) or bidirectional (both). In bidirectional mode the forward and reverse pulse sequences can be configured separately. A cycle count setting (set in the SetKCubeTrigParams message, lNumCycles parameter) allows the uni- or bidirectional position triggering sequence to be repeated a number of times.



Example for a move from 0 to 2000 position steps.

In forward direction: The first trigger pulse occurs at 1000 steps (StartPosFwd), the next trigger pulse occurs after another 500 steps (PosIntervalFwd), the stage then moves to 2000 steps.

In reverse direction: The next trigger occurs when the stage gets to 12oo steps.

Please note that position triggering can only be used on one TRIG port at a time.

The operation of the position triggering mode is described in more detail in the SetKCubeTrigParams message.

Trigger Polarity
The polarity of the trigger pulse is specified in the TrigPolarity parameters as follows:
0x01 The active state of the trigger port is logic HIGH 5V (trigger input and output on a rising edge).
0x02 The active state of the trigger port is logic LOW 0V (trigger input and output on a falling edge).

SET:
Command structure (32 bytes)
6 byte header followed by 26 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	1A	00	d|	s	SubMsgID		TrigChannel1		TrigChannel2	

12	13	14	15	16	17	18	19	20 to 31
Data								
Trig1Mode		Trig1Polarity		Trig2Mode		Trig2Polarity		Reserved

Data Structure:

field	description	format
SubMsg ID	The message ID (i.e. 17, 00) of the message containing the parameters	word
TrigChannel1	The drive channel that uses Trig 1 (I/O 1) as follows: Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
TrigChannel2	The drive channel that uses Trig 2 (I/O 2) as follows: Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
Trig1Mode	TRIG1 operating mode:	word
Trig1Polarity	The active state of TRIG1 (i.e. logic high or logic low) .	word
Trig2Mode	TRIG2 operating mode:	word
Trig2Polarity	The active state of TRIG2 (i.e. logic high or logic low) .	word
Reserved		6 words

Example:

TX C2, 08, 1A, 00, D0, 01, 17, 00, 01, 00, 02, 00, 02, 00, 01, 00, 10, 00, 01, 00, 00, 00

Header: C2, 08, 1A, 00, D0, 01: Set_KCube_TrigIOConfig, 16 byte data packet, d=D0 (i.e. 50 ORed with 80 i.e. generic USB device), s=01 (PC).

SubMsgID: 17,00 KCubeTrigIOConfig
TrigChannel1*: 01, 00*: Channel 1 to use Trig I/O 1
TrigChannel2: 02,00 Channel 2 to use Trig I/O 2
Trig1Mode – 02, 00 TrigIn_Relative Move
Trig1Polarity – 01,00 High
Trig2Mode – 10,00 Fwd Limit switch activated
Trig2Polarity – 01,00 High

REQ:

Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

GET:

Command structure 32 bytes

6 byte header followed by 26 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	10	00	d|	s	SubMsgID		Trig1Channel1		Trig1Channel2	

12	13	14	15	16	17	18	19	20 to 31
Data								
Trig1Mode		Trig1Polarity		Trig2Mode		Trig2Polarity		Reserved

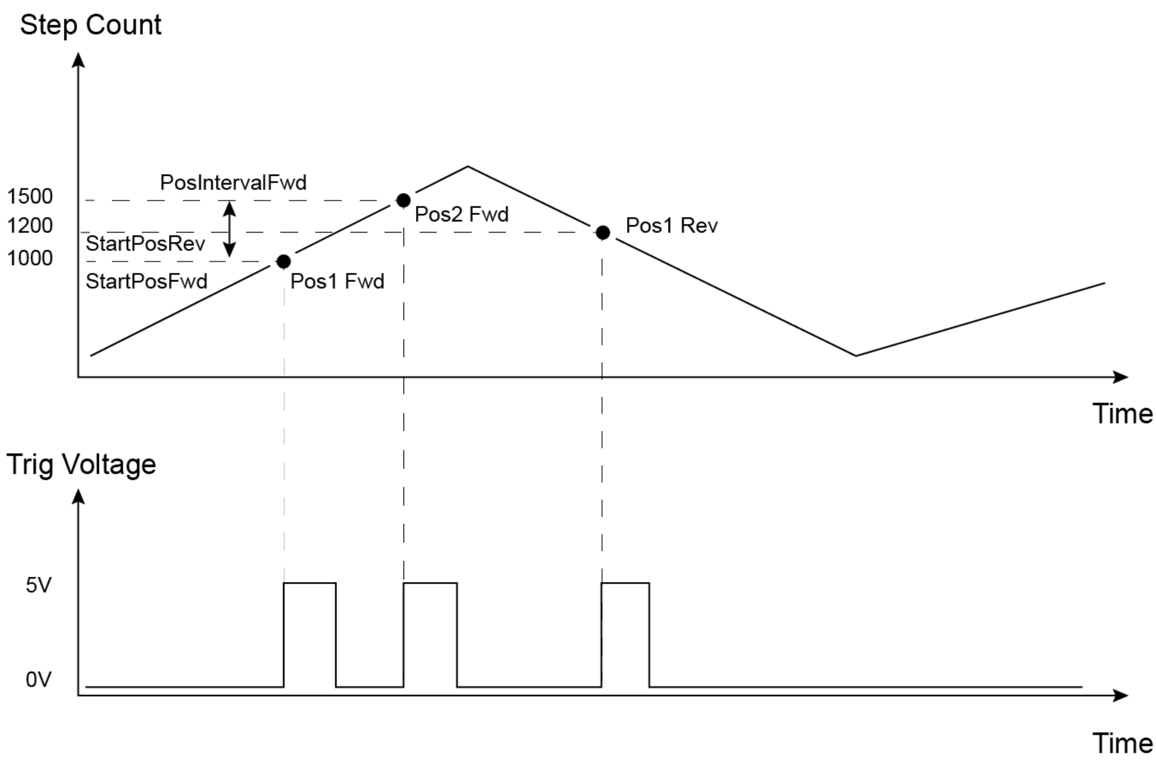
See SET message for structure.

Set/Request/Get_PZMOT_KCubeTrigParams (sub-message ID = 19)
Applicable only to KIM001 and KIM101 units

The KIM101 K-Cube inertial piezo motor controllers have two bidirectional trigger ports (I/O 1 and I/O 2) that can be set to be used as input or output triggers. This method sets operating parameters used when the triggering mode is set to a trigger out position steps mode by calling the PZMOT_KCubeTrigIOConfig message.
As soon as position triggering is selected on either of the TRIG ports, the port will assert the inactive logic state. As the stage moves in its travel range and the actual position matches the position set in the StartPosFwd parameter, the TRIG port will output its active logic state. The active state will be output for the length of time specified by the PulseWidth parameter, then return to its inactive state and schedule the next position trigger point at the "StartPosFwd value plus the value set in the fPosIntervalFwd parameter. Thus when this second position is reached, the TRIG output will be asserted to its active state again. The sequence is repeated the number of times set in the NumPulsesFwd parameter.
When the number of pulses set in the NumPulsesFwd parameter has been generated, the trigger engine will schedule the next position to occur at the position specified in the StartPosRev parameter. The same sequence as the forward direction is now repeated in reverse, except that the PosIntervalRev and NumPulsesRev parameters apply. When the number of pulses has been output, the entire forward-reverse sequence will repeat the number of times specified by NumCycles parameter. This means that the total number of pulses output will be NumCycles x (NumPulsesFwd + NumPulsesRev).

Once the total number of output pulses have been generated, the trigger output will remain inactive.

When a unidirectional sequence is selected, only the forward or reverse part of the sequence will be activated.



Example for a move from 0 to 20 mm and back.

In forward direction: The first trigger pulse occurs at 10 mm (StartPosFwd), the next trigger pulse occurs after another 5 mm (PosIntervalFwd), the stage then moves to 20 mm.

In reverse direction: The next trigger occurs when the stage gets to 12 mm.
Note that the position triggering scheme works on the principle of always triggering at the next scheduled position only, regardless of the actual direction of movement. If, for example, a position trigger sequence is set up with the forward start position at 10 mm, but initially the stage is at 15 mm, the first forward position trigger will occur when the stage is moving in the reverse direction. Likewise, if the stage does not complete all the forward position trigger points, the reverse triggering will not activate at all. For normal operation it is assumed that all trigger points will be reached during the course of the movement.

SET
Command structure (42 bytes)
6 byte header followed by 36 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	24	00	d	s	SubMsgID		Chan Ident		StartPosFwd			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

26	27	28	29	30	31	32	33	34	35	36	37
Data											
IntervalRev				NumPulsesRev				PulseWidth			

38	39	40	41
Data			
NumCycles			

Data Structure:

field	description	format
SubMsg ID	The message ID (i.e. 1900) of the message containing the parameters	word
Chan Ident	The channel being addressed as follows: Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
StartPosFwd -	When moving forward, this is the stage position [in position steps] to start the triggering sequence.	long
IntervalFwd	When moving forward, this is the interval [in position steps] at which to output the trigger pulses.	long
NumPulsesFwd	Number of output pulses during a forward move.	long
StartPosRev -	When moving backwards, this is the stage position [in position steps] to start the triggering sequence.	long
IntervalRev	When moving backwards, this is the interval [in position steps] at which to output the trigger pulses.	long
NumPulsesRev	Number of output pulses during a backwards move.	long
PulseWidth	Trigger output pulse width (from 1 µs to 100000 µs).	long
NumCycles	Number of forward/reverse move cycles.	long

REQ:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	Chan Ident	00	d	s

Example: Request the settings for the position trigger parameters

TX C1, 08, 01, 00, 50, 01

GET:
Response structure (42 bytes):
6 byte header followed by 36 byte data packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C2	08	24	00	d	s	SubMsgID		Chan Ident		StartPosFwd			

14	15	16	17	18	19	20	21	22	23	24	25
Data											
IntervalFwd				NumPulsesFwd				StartPosRev			

26	27	28	29	30	31	32	33	34	35	36	37
Data											
IntervalRev				NumPulsesRev				PulseWidth			

38	39	40	41
Data			
NumCycles			

For structure see SET message above.

Set/Request/Get_PZMOT_KCubeChanEnableMode (sub-message ID = 2B)
Applicable only to KIM001 and KIM101 units

In some applications (e.g. if the actuators are fitted to a 2-axis mirror mount), it may be advantageous to move two axes at the same time by moving the joystick diagonally.
The Channel 1 to 4 options allow each channel to be enabled and disabled individually.
The Channel Pair options are used to move two axes simultaneously (CH1 and 2, and CH3 and 4).

SET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
C0	08	04	00	d|	s	SubMsgID		Mode	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 2B00) of the message containing the parameters	word
Mode	The channel or channels to enable…. 00 - None, i.e. all channels disabled 01 - Channel 1 *The following parameter entries are applicable only to KIM101 units, they are not applicable to KIM001* 02 - Channel 2 03 - Channel 3 04 - Channel 4 05 - Channels 1 and 2 06 - Channels 3 and 4	word

Example: Enable channels 1 and 2:

TX C0, 08, 04, 00, A2, 01, 2B, 00, 05, 00,

Header: C0, 08, 04, 00, A2, 01: SetKCubeChanEnableMode, 4 byte data packet, Generic USB Device
SubMsg ID: 2B, 00
Mode: *05, 00:* Channels 1 and 2 enabled

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

GET:
Command structure (10 bytes)
6 byte header followed by 4 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						*Data*			
C0	08	04	00	d|	s	SubMsgID		Mode	

See SET for data structure.

Set/Request/Get_PZMOT_KCubeJogParams (sub-message ID = 2D
Applicable only to KIM001 and KIM101 units

This sub-message sets various jog parameters which define the speed and acceleration of moves initiated in the following ways:

by clicking the jog buttons on the GUI panel
by movng the joystick on the unit when 'Jog Mode' is selected.
via software using the MoveJog method.

It differs from the normal motor jog message in that there are two jog step sizes, one for forward and one for reverse. The reason for this is that due to the inherent nature of the PIA actuators going further in one direction as compared with another this will allow the user to potentially make adjustments to get fore and aft movement the same or similar.

Note. Drive parameters for motor moves are specified in the Set_PZMOT_DriveOPParams sub-message.

SET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	16	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JogStepSizeFwd				JogStepSizeRev				JogStepRate			

24	25	26	27
Data			
JogStepAccn			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 2D00) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
JogMode	Jog commands can be issued by calling the MoveJog method, or via the Motor Control GUI panel or by using the joystick on the hardware unit. When a jog command is received, if the jog mode is set to 1 (i.e. 'Continuous') the motor continues to move until the jog signal is removed (i.e. the jog button is released) when the motor will stop immediately. If the mode is set to '2' (i.e. Single Step) the motor moves by the step size specified in the JogStepSizeFwd and JogStepSizeRev parameters.	word
JogStepSizeFwd	A jog step consists of a number of drive pulses. This parameter specifies the number of pulses which make up a jog step when moving forwards in the range 1 to 2,000.	long

JogStepSizeRev	A jog step consists of a number of drive pulses. This parameter specifies the number of pulses which make up a jog step when moving backwards, in the range 1 to 2,000.	long
JogStepRate	The piezo motor moves by ramping up the drive voltage to the value set in the Set_TIM_DriveParameters sub-message and then dropping quickly to zero, then repeating. One cycle is termed a step. This parameter specifies the step rate (i.e. velocity) to move when a command is initiated. The step rate is specified in steps/sec, in the range 1 to 2,000	long
JogStepAccn	This parameter specifies the acceleration up to the step rate, in the range 1 to 100,000 cycles/sec/sec.	long

Example: Set the KIM Jog Parameters

TX C0,08,16,00,81,50, 2D,00,01,00,02,00,FA,00,00,00,F4,01,00,00,A0,86,01,00

Header: C0, 08, 16, 00, 81, 50: PZMOT_SET_PARAMS, 22 byte data packet, Generic USB Device.
SubMsgID: 2D, 00 *Set_KCubeJogParams*
ChanIdent: 01, 00 Channel 1
JogMode: 02, 00 Single Step Jog Mode
JogStepSizeFwd: FA. 00, 00, 00 250 steps
JogStepSizeRev: 04. 01, 00, 00 260 steps
JogStepRate: F4, 01, 00, 00 500 Steps/Sec (01F4)
JogStepAccn: A0, 86, 01, 00 10,000 Steps/Sec/Sec (0186A0

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

TX C1, 08, 01, 00, 50, 01,

GET:
Command structure (28 bytes)
6 byte header followed by 22 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	16	00	d|	s	SubMsgID		ChanIdent		JogMode	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
JogStepSize				JogStepRate				JogStepAccn			

See Set message for structure

Set/Request/Get_PZMOT_KCubeFeedbackSigParams (sub-message ID = 30
Applicable only to KIM001 and KIM101 units

The USER IO connector on the rear panel exposes two pairs of four digital inputs. These inputs can be used by a drive channel to receive a signal from the actuator being driven, either a differential QEP encoder feedback signal, or the FWD and REV limit switch signals. This sub message sets up the QEP/Limit switch selection for a specified channel.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C0	08	0A	00	d|	s	SubMsgID		ChanIdent		FBSignalMode	

12	13	14	15
Data			
EncoderConst			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 30,00) of the message containing the parameters	word
ChanIdent	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
FBSignalMode	This parameter sets the mode of the digital inputs, to receive either a feedback signal or a limit switch signal: 00 – DISABLED. The digital inputs are disabled 01 – LIMSWITCH. The inputs accept a signal when the limit switches are activated. **The following option is for future use and is not implemented at this time.** 02 – ENCODER. The inputs accept a feedback signal from the encoder in the actuator	word
EncoderConst	**This parameter is not implemented at this time.** If the *FBSignalMode* parameter above is set to Encoder 02, this parameter sets the calibration constant for converting encoder counts to real world units (mm or degrees) for the actuator being driven.	long

Example:

TX C0,08,0A,00,81,50, 30,00,01,00,02,00,FA,00,00,00,

Header: C0, 08, 0A, 00, 81, 50: PZMOT_SET_PARAMS, 10 byte data packet, Generic USB Device.
SubMsgID: 30, 00 *Set_KCubeFBSigParams*
ChanIdent: 01, 00 Channel 1
FBSignalMode: 02, 00 Encoder Signal
EncoderConst: FA. 00, 00, 00 250 steps/mm

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

TX C1, 08, 01, 00, 50, 01,

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
C2	08	0A	00	d|	s	SubMsgID		ChanIdent		FBSignalMode	

12	13	14	15
Data			
EncoderConst			

See Set message for structure

Set/Request/Get_PZMOT_KCubeMoveRelativeParams (sub-message ID = 32)
Applicable only to KIM001 and KIM101 units

Used to set the relative distance moved when the trigger mode is set to TRIGIN_RELMOVE in the PZMOT_KCubeTrigIOConfig (17) sub-message.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	08	00	d|	s	SubMsgID		Channel		RelDistance			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3200) of the message containing the parameters	word
Channel	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
RelDistance	The relative distance to move (in position steps, negative or positive) when the trigger mode is set to TRIGIN_RELMOVE (see PZMOT_KCubeTrigIOConfig)	long

Example:

TX C0, 08, 08, 00, 81, 50, 32, 00, 01, 00, E8, 03

Header: C0, 08, 08, 00, 81, 50: Set KIM MoveRelativeParams, 8 byte data packet, Generic USB Device
SubMsg ID: 32, 00
Channel: 01, 00 Channel 1
RelDistance: *E8, 03 i.e. 1,000 steps*

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	08	00	d|	s	SubMsgID		Channel		RelDistance			

See SET for data structure.

Set/Request/Get_PZMOT_KCubeMoveAbsoluteParams (sub-message ID = 34)
Applicable only to KIM001 and KIM101 units

Used to set the absolute distance moved when the trigger mode is set to TRIGIN_ABSMOVE in the PZMOT_KCubeTrigIOConfig (17) sub-message.

SET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	08	00	d|	s	SubMsgID		Channel		AbsDistance			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3400) of the message containing the parameters	word
Channel	The channel to be addressed. Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
AbsDistance	The absolute distance to move (in position steps) when the trigger mode is set to TRIGIN_ABSMOVE (see PZMOT_KCubeTrigIOConfig)	long

Example:

TX C0, 08, 08, 00, 81, 50, 34, 00, 01, 00, 10,27

Header: C0, 08, 08, 00, 81, 50: Set KIM MoveAbsoluteParams, 8 byte data packet, Generic USB Device
SubMsg ID: 34, 00
Channel: 01, 00 Channel 1
AbsDistance: *10, 27 i.e. 10,000 steps*

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	01	00	d	s

GET:
Command structure (14 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
header						*Data*							
C0	08	08	00	d|	s	SubMsgID		Channel		AbsDistance			

See SET for data structure

Set/Request/Get_PZMOT_ CloseLoopParams (sub-message ID = 39)
Applicable to PDXC2

This sub-message sets the closed loop operating parameters.

SET:
Command structure (30 bytes)
6 byte header followed by 24 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	18	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
RefSpeed				Proportional				Integral	

20	21	22	23	24	25	26	27	28	29
Data									
Integral		Derivative				Acceleration			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
RefSpeed	Set the reference speed for stage, from 0 - 20000000 nm/s	Long
Proportional	Set the Kp of PID, from 0 - 32767	Long
Integral	Set the Ki of PID, from 0 - 32767	Long
Derivative	Set the Kd of PID, from 0 - 32767	Long
Acceleration	Set the acceleration for trajectory control, from 0 - 1000000000 nm/s^2	Long

Example: Set the closed loop parameters
TX C0 ,08 ,18 ,00 ,d0 ,01 ,39 ,00 ,01 ,00 ,40,42 ,0F ,00 ,E8 ,03 ,00 ,00 ,E8 ,03 ,00 ,00 ,E8 ,03 ,00 ,00 , 40,42 ,0F ,00

Header: C0, 08, 18, 00, D0, 01: PZMOT_SET_PARAMS, 24-byte data packet, Generic USB Device. *SubMsgID: 39, 00 Set_PZMOT_CloseLoopParams*
ChanIdent: 01, 00 Channel 1
RefSpeed*:* 40,42 ,0F ,00 1000000 nm/s
Proportional*:* E8 ,03 ,00 ,00 1000
Integral: E8 ,03 ,00 ,00 1000
Derivative: E8 ,03 ,00 ,00 1000
Acceleration: 40,42 ,0F ,00 1000000 nm/s^2

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 39, 01, 50, 01,

GET:
Command structure (30 bytes)
6 byte header followed by 24 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	18	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
RefSpeed				Proportional				Integral	

20	21	22	23	24	25	26	27	28	29
Data									
Integral		Derivative				Acceleration			

Set/Request/Get_PZMOT_ Etmconfig (sub-message ID = 3B)
Applicable to PDXC2

This sub-message sets the external trigger operating mode.

SET:
Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	06	00	d	s	SubMsgID		ChanIdent	

10	11
Data	
ExtTrigMode	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3B00) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
ExtTrigMode	0: Manual Mode, 1: Analog In with Rising trigger edge mode. 2: Analog In with Failing trigger edge mode. 3: Fixed-Step Distance with Rising trigger edge mode. 4: Fixed-Step Distance with Edging trigger edge mode. 5: Two-Fixed-Position with Rising	Word

	trigger edge mode. 6: Two-Fixed-Position with Edging trigger edge mode. Others: Manual Mode,	

Example: Set the External Trigger Mode to analog in with Rising trigger edge mode
TX C0 ,08 ,06 ,00 ,d0 ,01 ,3B ,00 ,01 ,00 ,01 ,00

Header: C0, 08, 06, 00, D0, 01: PZMOT_SET_PARAMS, 6-byte data packet, Generic USB Device. *SubMsgID: 3B, 00 Set_PZMOT_EtmConfig*
ChanIdent: 01, 00 Channel 1.
ExtTrigMode*:* 01,00 Analog in with Rising trigger edge mode.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 3B, 01, 50, 01,

GET:
Command structure (6 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	06	00	d	s	SubMsgID		ChanIdent	

10	11
Data	
ExtTrigMode	

See Set message for structure

Set/Request/Get_PZMOT_ EtmParams (sub-message ID = 3C)
Applicable to PDXC2

This sub-message sets the external trigger operating parameter.
SET:
Command structure (50 bytes)
6 byte header followed by 44 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	2C	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
RiseFixStep				FailFixStep				RisePos1	

20	21	22	23	24	25	26	27	28	29
Data									
RisePos1		FailPos1				RisePos2			

30	31	32	33	34	35	36	37	38	39
Data									

FailPos2				InAnalogGain				InAnalogOffset	

40	41	42	43	44	45	46	47	48	49
Data									
InAnalogOffset		OutAnalogGain				OutAnalogOffset			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3C00) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
RiseFixStep	Rise Fixed-Step Distance, the unit is nm. From -10000000 to +10000000	Long
FailFixStep	Fail Fixed-Step Distance, the unit is nm. From -10000000 to +10000000	Long
RisePos1	First position of rise trigger, the unit is nm. From -10000000 to +10000000	Long
FailPos1	First position of fail trigger, the unit is nm. From -10000000 to +10000000	Long
RisePos2	Second position of rise trigger, the unit is nm. From -10000000 to +10000000	Long
FailPos2	Second position of fail trigger, the unit is nm. From -10000000 to +10000000	Long
InAnalogGain	Analog In gain, from 0-32768	Float
InAnalogOffset	Analog In offset, from -10000 to 10000 (mv)	Long
OutAnalogGain	Analog out offset, from 0-32768	Float
OutAnalogOffset	Analog out offset , from -10000 to 10000 (mv)	Long

Example: Set the RiseFixStep and FailFixStep value is 1000nm, First position of rise trigger is 1000000nm. Second position of rise trigger is 10000000nm. First position of fail trigger is -1000000mm. Second position of fail trigger is -10000000mm. InAnalogGain is 1, InAnalogOffset is 1000mv , OutAnalogGain is 1, OutAnalogOffset is 500mv.

TX C0 ,08 ,2C ,00 ,d0 ,01 ,3C ,00 ,01 ,00 ,E8 ,03 ,00 ,00 ,E8 ,03 ,00 ,00 ,E8 ,03 ,00 ,00 ,40 ,42 ,0F ,00 ,80 ,96 ,98 ,00 ,C0 ,BD ,F0 ,FF ,00 ,00 ,80 ,3F ,E8 ,03 ,00 ,00 ,00 ,00 ,C0 ,3F ,F4 ,01 ,00 ,00

Header: C0, 08, 2C, 00, D0, 01: PZMOT_SET_PARAMS, 44-byte data packet, Generic USB Device. SubMsgID: 3C, 00 Set_PZMOT_EtmParams
ChanIdent: 01, 00 Channel 1
RiseFixStep: E8 ,03 ,00 ,00 1000nm
FailFixStep: ,E8 ,03 ,00 ,00 1000nm
RisePos1: E8 ,03 ,00 ,00 First position of rise trigger is 1000nm.
FailPos1: 40 ,42 ,0F ,00 First position of fail trigger is 1000nm.

RisePos2: 80 ,96 ,98 ,00 Second position of rise trigger is 1000000nm.
FailPos2: C0 ,BD ,F0 ,FF Second position of fail trigger is -1000000nm.
InAnalogGain: 00 ,00 ,80 ,3F InAnalogGain is 1.0
InAnalogOffset: E8 ,03 ,00 ,00 InAnalogOffset is 1000mv
OutAnalogGain: 00 ,00 ,C0 ,3F OutAnalogGain is 2.0
OutAnalogOffset: F4 ,01 ,00 ,00 OutAnalogOffset is 500mv

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 3C, 01, 50, 01,

GET:
Command structure (44 bytes)
6-byte header followed by 44-byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	2C	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
RiseFixStep				FailFixStep				RisePos1	

20	21	22	23	24	25	26	27	28	29
Data									
RisePos1		FailPos1				RisePos2			

30	31	32	33	34	35	36	37	38	39
Data									
FailPos2				InAnalogGain				InAnalogOffset	

40	41	42	43	44	45	46	47	48	49
Data									
InAnalogOffset		OutAnalogGain				OutAnalogOffset			

See Set message for structure

Set/Request/Get_PZMOT_ TPOS (sub-message ID = 3D)
Applicable to PDXC2

Used to query target position when in external trigger mode, whose signal Analog In reflects the value.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 3D, 01, 50, 01,

GET:
Command structure (8 bytes)
6-byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
TargetPos			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3D00) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
TargetPos	Target position when in external trigger mode, whose signal Analog In reflect the value. Range from -10000000 to 10000000 nm.	Long

Example: Get the analog target value
TX C2 ,08 ,08 ,00 ,d0 ,01 ,3D ,00 ,01 ,00 ,10 ,27 ,00 ,00

Header: C2, 08, 08, 00, D0, 01: PZMOT_GET_PARAMS, 8 byte data packet, Generic USB Device. *SubMsgID: 3D, 00 Get_PZMOT_TPOS*
ChanIdent: 01, 00 Channel 1.

TargetPos*:* 10 ,27 , 00 , 00 The analog target value is 10000 nm.

Set/Request/Get_PZMOT_ EthrenetParams (sub-message ID = 3F)
Applicable to PDXC2

Set the ethernet operating IP address parameters.
SET:
Command structure (24 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	12	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
GetIpSel		IpAddr				MaskAddr			

20	21	22	23
Data			
GatewayAddr			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
GetIpSel	0x0:Get IP address from DHCP server. 0x1:Get IP address from Parameters.	Word
IpAddr	Set the Ip address.	char[N]
MaskAddr	Set the Mask Address.	char[N]
GatewayAddr	Set the Gateway Address.	char[N]

Example: Set the GetIpsel to 0x0(Get ip from DHCP Server), IpAddr to 192.168.1.10 , MaskAddr to 255.255.255.0, GatawayAddr to 192.168.1.1 .

TX C0 ,08 ,12 ,00 ,d0 ,01 ,3F ,00 ,01 ,00 ,00 ,00 ,0A ,01 ,9E ,C0 ,00 ,FF ,FF ,FF ,01 ,01 ,9E ,C0

Header: C0, 08, 12, 00, D0, 01: PZMOT_SET_PARAMS,18 byte data packet, Generic USB Device. *SubMsgID: 3F, 00 Set_PZMOT_EthernetParams*
ChanIdent: 01, 00 Channel 1.
GetIpSel*:* 00,00 Get ip from DHCP Server.
IpAddr: 0A ,01 ,9E ,C0 IpAddr is 192.168.1.10
MaskAddr: 00 ,FF ,FF ,FF MaskAddr is 255.255.255.0
GatewayAddr: 01 ,01 ,9E ,C0 GatawayAddr is 192.168.1.1

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 3F, 01, 50, 01,

GET:
Command structure (18 bytes)
6 byte header followed by 18 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	12	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
GetIpSel		IpAddr				MaskAddr			

20	21	22	23
Data			
GatewayAddr			

See Set message for structure

Set/Request/Get_PZMOT_ CURRENT_POS (sub-message ID = 40)
Applicable to PDXC2

Return current position in unit of nm/steps.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 40, 01, 50, 01,

GET:
Command structure (8 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
CurrentPos			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3D00) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
TargetPos	If the operation mode is close loop, The current position in unit of nm. Range from -10000000 to +10000000 nm, and if the operation mode is open loop, The current position in unit of steps, Range from -10000000 to +10000000 steps	Long

Example: Get the current position value
TX: C2 ,08 ,08 ,00 ,d0 ,01 ,40 ,00 ,01 ,00 ,10 ,27 ,00 ,00

Header: C2, 08, 08, 00, D0, 01: PZMOT_GET_PARAMS, 8 byte data packet, Generic USB Device. *SubMsgID: 40, 00 Get_PZMOT_CURRENT_POS*
ChanIdent: 01, 00 Channel 1.
TargetPos*:* 10 ,27 , 00 , 00 Get the current position value is 10000 nm in closed loop mode or 10000 steps in open loop mode.

Set/Request/Get_PZMOT_ AMD (sub-message ID = 43)
Applicable to PDXC2

Used to enable/disable the detection of abnormal move.

SET:
Command structure (6 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	06	00	d	s	SubMsgID		ChanIdent	

10	11
Data	
AmdEn	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3B00) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
AmdEn	used to enable/disable the detection of abnormal move, AmdEn = 0x0 diable , AmdEn = 0x1 Enable .	Word

Example: Set the detection for abnormal move to enable.
TX C0 ,08 ,06 ,00 ,d0 ,01 ,43 ,00 ,01 ,00 ,01 ,00

Header: C0, 08, 06, 00, D0, 01: PZMOT_SET_PARAMS, 6-byte data packet, Generic USB Device. *SubMsgID: 43, 00 Set_PZMOT_AMD*
ChanIdent: 01, 00 Channel 1.
AmdEn*:* 01,00 enable abnormal move check.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 43, 01, 50, 01,

GET:
Command structure (6 bytes)
6-byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	06	00	d	s	SubMsgID		ChanIdent	

10	11
Data	

AmdEn

See Set message for structure

Set/Request/Get_PZMOT_JogParams (sub-message ID = 44)
Applicable to PDXC2

Used to set jog parameters.

SET:
Command structure (36 bytes)
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	1E	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
JogMode		OpenLoopStepSize				OpenLoopStepRate			

20	21	22	23	24	25	26	27	28	29
Data									
OpenLoopAcceleration				ClosedLoopStepSize				ClosedLoopSpeed	

30	31	32	33	34	35
Data					
ClosedLoopSpeed		ClosedLoopAcceleration			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
JogMode	used to set the jog mode , 0x1 continues , 0x1 single step.	Word
OpenLoopStepSize	A jog step consists of several drive pulses. This parameter specifies the number of pulses which make up a jog step, in the range 1 to 2,0000.	Long
OpenLoopStepRate	Set the move frequency of output pulse.range is from 800 to 20000 Steps/s.	Long
OpenLoopAcceleration	This parameter specifies the acceleration up to the step rate, in the range 1 to 20000 Steps/s^2	Long
ClosedLoopStepSize	Set the jog step size for closed loop mode, unit is nm, range from 0 - 2000000 nm.	Long
ClosedLoopSpeed	Set the reference speed for stage, from 0 - 20000000 nm/s	Long
ClosedLoopAcceleration	Set the acceleration for trajectory	Long

	control, from 0 - 1000000000 nm/s^2	

Example: Set the jog parameters.
TX: C0 ,08 ,1E ,00 ,d0 ,01 ,44 ,00 ,01 ,00 ,01 ,00 ,10 ,27 ,00 ,00 ,10 ,27 ,00 ,00 ,10 ,27 ,00 ,00 ,40 ,42 ,0F ,00 ,80 ,96 ,98 ,00 , 80 ,96 ,98 ,00

Header: C0, 08, 1E, 00, D0, 01: PZMOT_SET_PARAMS, 30 byte data packet, Generic USB Device. *SubMsgID: 44, 00 Set_PZMOT_JogParams*
ChanIdent: 01, 00 Channel 1.
JogMode*:* 01, 00 Continues mode.
OpenLoopStepSize*:* 10 ,27 ,00 ,00 1000 steps
OpenLoopStepRate: 10 ,27 ,00 ,00 1000 steps/s
OpenLoopAcceleration: 10 ,27 ,00 ,00 1000 steps/s^2
ClosedLoopStepSize: 40 ,42 ,0F ,00 1000000 nm
ClosedLoopSpeed: 80 ,96 ,98 ,00 10000000 nm/s
ClosedLoopAcceleration: 80 ,96 ,98 ,00 10000000 nm/s^2

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 44, 01, 50, 01,

GET:
Command structure (36 bytes)
6 byte header followed by 30 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	1E	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15	16	17	18	19
Data									
JogMode		OpenLoopStepSize				OpenLoopStepRate			

20	21	22	23	24	25	26	27	28	29
Data									
OpenLoopAcceleration				ClosedLoopStepSize				ClosedLoopSpeed	

30	31	32	33	34	35
Data					
ClosedLoopSpeed		ClosedLoopAcceleration			

See Set message for structure

Set/Request/Get_PZMOT_AmpOutParams (sub-message ID = 45)
Applicable to PDXC2

Used to set the amplifier output parameter.

SET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	0A	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15
Data					
OutMode		ForwardAmpLevel		BackAmpLevel	

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
OutMode	Reserved.	Word
ForwardAmpLevel	set output forward amplitude level, defined by x, x ranges from 10 to 100 percent of max amplitude level. In open loop mode, can change speed by set AmpLevel value.	Word
BackAmpLevel	set output backward amplitude level, defined by x, x ranges from 10 to 100 percent of max amplitude level. In open loop mode, can change speed by set AmpLevel value.	Word

Example: Set forward amplitude level to 100 percent of max amplitude level and backward amplitude level to 50 percent of max amplitude level.
TX: C0 ,08 ,0A ,00 ,d0 ,01 ,45 ,00 ,01 ,00 ,00 ,00 ,64 ,00 ,32 ,00

Header: C0, 08, 0A, 00, D0, 01: PZMOT_SET_PARAMS, 10 byte data packet, Generic USB Device. *SubMsgID: 45, 00 Set_PZMOT_AmpOutParams*
ChanIdent: 01, 00 Channel 1.
OutMode*:* 00, 00 Reserved, no used.
ForwardAmpLevel*:* 64 ,00 100 percent of max amplitude level for forward.
BackAmpLevel: 32 ,00 50 percent of max amplitude level for forward.

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					

C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 45, 01, 50, 01,

GET:
Command structure (16 bytes)
6 byte header followed by 10 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	0A	00	d	s	SubMsgID		ChanIdent	

10	11	12	13	14	15
Data					
OutMode		ForwardAmpLevel		BackAmpLevel	

See Set message for structure

Set/Request/Get_PZMOT_OpenMoveParams (sub-message ID = 46)
Applicable to PDXC2

Used to the open loop move target steps value.

SET:
Command structure (8 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
StepSize			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
StepSize	Set the target absolute steps. From -10000000 to +10000000	Long

Example: Set target absolute steps to 10000.

TX: C0 ,08 ,08 ,00 ,d0 ,01 ,46 ,00 ,01 ,00 ,10 ,27 ,00 ,00

Header: C0, 08, 08, 00, D0, 01: PZMOT_SET_PARAMS, 8 byte data packet, Generic USB Device. *SubMsgID: 46, 00 Set_PZMOT_OpenMoveParams*
ChanIdent: 01, 00 Channel 1.
StepSize*:* 10 ,27 ,00 ,00 Target absolute steps to 10000

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 46, 01, 50, 01,

GET:
Command structure (8 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
StepSize			

See Set message for structure

Set/Request/Get_PZMOT_CloseMoveParams (sub-message ID = 47)
Applicable to PDXC2

Used to set the closed loop target position.

SET:
Command structure (8 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C0	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
DePos			

Data Structure:

field	description	format
SubMsgID	The message ID (i.e. 3900) of the message containing the parameters	Word
ChanIdent	The channel being addressed	Word
DePos	Set the desired position. The range is from -1000000 to +1000000 nm,	Long

Example: Set target position to 1000000nm.

TX: C0 ,08 ,08 ,00 ,d0 ,01 ,47 ,00 ,01 ,00 ,40 ,42 ,0F ,00

Header: C0, 08, 08, 00, D0, 01: PZMOT_SET_PARAMS, 8 byte data packet, Generic USB Device. *SubMsgID: 47, 00 Set_PZMOT_CloseMoveParams*

ChanIdent: 01, 00 Channel 1.
DePos*:* 40 ,42 ,0F ,00 Target position is 1000000 nm

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
C1	08	SubMsgID	ChanIdent	d	s

TX C1, 08, 47, 01, 50, 01,

GET:
Command structure (8 bytes)
6 byte header followed by 8 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9
header						Data			
C2	08	08	00	d	s	SubMsgID		ChanIdent	

10	11	12	13
Data			
DePos			

See Set Message for Structure

MGMSG_PZMOT_MOVE_ABSOLUTE MGMSG_PZMOT_SET_PARAMS 0x08D4

Function: Used to start a move to a position specified as the number of steps away from the zero position. The move will be executed using the parameters set in the TIM_Set_DriveOPParams sub-message.

Command structure (12 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D4	08	06	00	d|	s	Chan Ident		AbsPosition			

Data Structure:

field	description	format
Chan Ident	The channel being addressed Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
AbsPosition	The distance to move, relative to the zero position, specified in number of steps.	long

Example: Set an absolute move to 100 steps

Tx D4,08,06,00,D0,01,01,00,64,00,00,00

Header: D4,08,06,00,D0,01: PZMOT_MOVE_ABSOLUTE, 6 byte data packet, Generic USB Device.
ChanIdent: 01, 00 Channel 1
AbsPosition: 64. 00, 00, 00 100 steps (H64) from the zero position

On completion of the move, a Move_Completed message will be sent.

MGMSG_PZMOT_MOVE_COMPLETED 0x08D6

Function: No response on initial message, but upon completion of the absolute move sequence, the controller sends a "move completed" message:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
D6	08	0E	00	d|	s	Chan Ident		AbsPosition			

12	13	14	15	16	17	18	19
Data							
EncCount				Status Bits			

Data Structure:

field	description	format
Chan Ident	The channel being addressed Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
AbsPosition	The distance moved, relative to the zero position, specified in number of steps.	long

Example: Send message that move to 100 steps is complete

RX D6,08,0E,00,81,50,01,00,64,00,00,00,00,00,00,00,00,00,00,00

Header: D6,08,0E,00,81,50: PZMOT_MOVE_COMPLETE, 14 byte data packet, Generic USB Device.
ChanIdent: 01, 00 Channel 1
AbsPosition: 64. 00, 00, 00 100 steps (H64) from the zero position
EncCount: Not Used
StatusBits: Not Used

MGMSG_PZMOT_MOVE_JOG 0x08D9

Function: Used to start a jog move. The move will be executed using the parameters set in the TIM_Set_JogParameters sub-message.

Command structure (6 bytes)
6 byte header followed by 6 byte data packet as follows:

0	1	2	3	4	5
header					
D9	08	ChanIdent	JogDir	d	s

Channel Idents
0x01 channel 1
0x02 channel 2
0x03 channel 3
0x04 channel 4

JogDir
0x01 Forward
0x02 Reverse

Example
TX D9,08,01,01,50,01

On completion of the move, a Move_Completed message will be sent.

MGMSG_PZMOT_REQ_STATUSUPDATE 0x08E0
MGMSG_PZMOT_GET_STATUSUPDATE 0x08E1

Function: This message is returned 10 times a second, when status update messages have been requested using the MGMSG_HW_START_UPDATEMSGS function.

GET:
Status update messages are received with the following format:-

Response structure (62 bytes)
6 byte header followed by 56 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
E1	08	38	00	d|	s	Chan Ident		Position1			

12	13	14	15	16	17	18	19
Data							
EncCount1				Status Bits1			

Data Structure:

field	description	format
Chan Ident	The channel being addressed Chan 1 = 1, Chan 2 = 2, Chan 3 = 4, Chan 4 = 8	word
Position1	The position count for channel 1.	long
EncCount1	Not Used.	long
StatusBits1	The status bits for channel 1 – see below.	dword

The remaining 42 bytes for channel 2 to channel 4 are the same as for channel 1

bit mask	meaning
0x00000001	forward (CW) hardware limit switch is active
0x00000002	reverse (CCW) hardware limit switch is active
0x00000010	in motion, moving forward (CW)
0x00000020	in motion, moving reverse (CCW)
0x00000040	in motion, jogging forward (CW)
0x00000080	in motion, jogging reverse (CCW)
0x00000100	motor connected
0x00000200	in motion, homing
0x00000400	homed (homing has been completed)
0x00100000	digital input 1
0x10000000	power OK
0x20000000	active
0x40000000	error
0x80000000	channel enabled
0x01000000	Excessive current
0x02000000	Excessive temperature
0x04000000	Abnormal stage movement
0x08000000	Wrong stage detected

MGMSG_PZMOT_ACK_STATUSUPDATE 0x08E2

Only Applicable If Using USB COMMS. Does not apply to RS-232 COMMS

Function: If using the USB port, this message called “server alive” is sent by the server to the controller after 10 status update message.
The controller keeps track of the number of "status update" type of messages (e.g.move complete message) and if it has sent 10 of these without the server sending a "server alive" message, it will stop sending any more "status update" messages.
This function is used by the controller to check that the PC/Server has not crashed or switched off. There is no response.

Structure (6 bytes):

0	1	2	3	4	5
header only					
E2	08	00	00	d	s

TX E2, 08, 00, 00, 50, 01

MPC220 and MPC320 Control Messages

Introduction

The functionality for the MPC220 and MPC320 Polarization Controllers is accessed via the POL Control Object, and provides the functionality required for a client application to control a number of Controller units.
Every hardware unit is factory-programmed with a unique 8-digit serial number. This serial number is key to operation of the Thorlabs Server software and is used by the Server to enumerate and communicate independently with multiple hardware units connected on the same USB bus.
The serial number must be allocated using the HWSerialNum property, before an ActiveX control can communicate with the hardware unit. This can be done at design time or at run time.
The methods of the Polarization Controller can then be used to perform activities such as setting the home position or setting the jog step size.

Note. The channel being addressed must be enabled by calling the Set_ChanEnableState method, before the following methods can be used.

MGMSG_POL_SET_PARAMS 0x0530
MGMSG_ POL_REQ_PARAMS 0x0531
MGMSG_ POL_GET_PARAMS 0x0532

Function: This generic parameter set/request message is used to control the functionality of the MPC220 and MPC320 polarization controllers. The specific parameters to control are identified below.

SET:

Command structure (18 bytes)

6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
30	05	0A	00	d|	s	Not Used		Velocity		HomePosition	

12	13	14	15	16	17
Data					
JogStep1		JogStep2		JogStep3	

Data Structure:

field	description	format
Velocity	The velocity of motion when a move command is received. The setting is global (i.e. applies to all 3 paddles), and is set in the range 10% to 100% of the max 400°/s.	word
HomePosition	The home position is global (i.e. applies to all 3 paddles). It is set in encoder counts and is usually set to 0 but it can be set anywhere in the range 0 to 1370 (0 to 170°) depending on the application requirements.	word
JogStep1	The size of step to be performed on paddle No. 1, each time the MoveJog command is called. Step size is set in encoder counts in the range 0 to 1370 (0 to 170°).	word
JogStep2	The size of step to be performed on paddle No. 2, each time the MoveJog command is called. Step size is set in encoder counts in the range 0 to 1370 (0 to 170°).	word
JogStep3	The size of step to be performed on paddle No. 3, each time the MoveJog command is called. Step size is set in encoder counts in the range 0 to 1370 (0 to 170°).	word

Example: Set the polarization controller parameters as follows:
Velocity 50%
Home Position 0
Jog step size 3°for each paddle

TX 30, 05, 0C, 00, D0, 01, 00, 00, 32, 00, 00, 00, 19, 00, 19, 00, 19, 00

Header: 30, 05, 0C, 00, D0, 01: Set Params, 12 byte data packet, Generic USB Device
Not Used: 00, 00
Velocity: 32, 00 50%
Home Position: 00, 00 0°
JogStep1: 19, 00 25 encoder counts (3°)
JogStep2: 19, 00 25 encoder counts (3°)
JogStep3: 19, 00 25 encoder counts (3°)

REQUEST:
Command structure (6 bytes):

0	1	2	3	4	5
header only					
31	05	00	00	d	s

GET:
Response structure (12 bytes)
6 byte header followed by 12 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
32	05	0A	00	d|	s	Not Used		Velocity		HomePosition	

12	13	14	15	16	17
Data					
JogStep1		JogStep2		JogStep3	

CT1P Control Messages

Introduction

The functionality for the CT1P Piezo Cage Translator is accessed via the Thorlabs Piezo Control Object, and provides the functionality required for a client application to control a number of Controller units.
Every hardware unit is factory programmed with a unique 8-digit serial number. This serial number is key to operation of the Thorlabs Server software and is used by the Server to enumerate and communicate independently with multiple hardware units connected on the same USB bus.
The serial number must be allocated using the HWSerialNum property, before an ActiveX control can communicate with the hardware unit. This can be done at design time or at run time.
The functions of the Piezo Control can then be used to perform activities such as setting the drive voltage, setting the jog step voltage and setting device control panel parameters. The functions applicable to the CT1P are listed here.

For details on the use of the CT1P unit, refer to the handbook available to download from www.thorlabs.com.

MGMSG_PZ_REQ_PIDCRITERIA 0x0699
MGMSG_ PZ_GET_PIDCRITERIA 0x069A
MGMSG_ PZ_SET_PIDCRITERIA 0x069B

In the main, the CT1P uses general Piezo and Strain Gauge unit functions and these are listed here. The following function is applicable only to the CT1P Piezo Cage Translator.

Function: The unit has two groups of PID settings. During normal moves, the unit uses a group of PID settings that are generally chosen for a fast response. When approaching the desired target (set in the Target Error Window parameter) then the CT1P uses PID settings which incorporate a different set of values chosen for low noise and stability. The Target Error Window defines how close to the target position the device is before switching PID groups.

SET:
Command structure (20 bytes)
6 byte header followed by 14 byte data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
header						*Data*					
9B	06	0A	00	d|	s	Channel		Index		Priority	

12	13	14	15	16	17	18	19
Data							
TargetErrWin		Reserved		Reserved		Reserved	

Data Structure:

field	description	format
Channel	Byte 1 sets the channel but for the CT1P is always set to 1, byte 2 sets the criteria as follows: 40 – Strain Gauge Near Target 41 – Encoder Near Target	word
Index	When operating in Closed Loop mode, the proportional, integral and derivative (PID) constants can be used to fine tune the behaviour of the feedback loop to changes in the output voltage or position. While closed loop operation allows more precise control of the output position, feedback loops need to be adjusted to suit the different types of operation. Therefore, the unit has two groups of PID settings. During normal moves, the unit uses the PID settings ID 0 that are generally chosen for a fast response. When approaching the desired target (set in the Target Error Window field) then the CT1P uses the PID settings ID1, which incorporates a different set of values chosen for low noise and stability. The actual PID .values are set using the SET_PPC_PIDCONSTS message, which must be called twice, once for each Index parameter value.	word
Priority	For Future Use.	word

TargetErrorWindow	When the unit is approaching the requested position, the device switches to using the ID1 set of PID parameters. The Target Error Window defines how close to the target position the device gets before switching parameter sets.	word
Reserved		word
Reserved		word
Reserved		word

MGMSG_KPC_SET_KCUBETRIGIOCONFIG 0x07FC
MGMSG_KPC_REQ_KCUBETRIGIOCONFIG 0x07FD
MGMSG_KPC_GET_KCUBETRIGIOCONFIG 0x07FE

Function: This message is used to configure the I/O connectors on the KPC101 controller. The message contains configuration settings for the two digital I/O ports and the analogue MONITOR output. Note that the EXT IN analogue input is configured using a separate command (MGMSG_KPC_SET_IOSETTINGS).

SET:
Command structure (38 bytes)
6 byte header followed by 32 bytes data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
FC	07	20	00	d |0x80	s	Channel ident		Trig 1 Mode		Trig 2 Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig 2 Mode		Trig 2 Polarity		Strain Gauge Lower Limit				Strain Gauge Upper Limit			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
SmSamples		Monitor Mode		Monitor Freq		Monitor Sw Val		Reserved		Reserved	

36	37
Data	
Reserved	

Data Structure:

field	description	format
Channel indent	The channel being addressed	word
	I/O connector #1 configuration for trigger functionality. Values defined are: 0 = Disabled, the state of the I/O port has no effect on the piezo controller. 1 = General purpose logic input (read through status bits) 2 = Input trigger for voltage/position step up (uses lJSVoltStep from KPZMMIPARAMS) for step size 3 = Input trigger for voltage/position step down (uses lJSVoltStep from KPZMMIPARAMS) 10 = General purpose logic output (set using	word

Trigger I/O #1 Mode	MGMSG_MOD_SET_DIGOUTPUTS). 11 = Output, set to active level when strain gauge reading < lower limit, defined in this command 12 = Output, set to active level when strain gauge reading > lower limit 13 = Output, set to active level when strain gauge reading < upper limit 14 = Output, set to active level when strain gauge reading > upper limit 15 = Output, set to active level when strain gauge reading > lower limit and < upper limit 16 = Output, set to active level when strain gauge reading < lower limit and > upper limit Any other value = reserved, unused	
Trigger I/O #1 Polarity	I/O connector #1 polarity 1 = Active HIGH 2 = Active LOW Any other value = reserved, unused	word
Trigger I/O #2 Mode	The same values as for Trigger I/O #1, described above	word
Trigger I/O #1 Polarity	The same values as for Trigger I/O #1, described above	word
Strain Gauge Lower Limit	Lower limit for strain gauge trigger thresholds, 16-bit signed value of 0 to 32767 mapping to the zero to maximum position.	long
Strain Gauge Upper Limit	Upper limit for strain gauge trigger thresholds, 16-bit signed value of 0 to 32767 mapping to the zero to maximum position.	long
Smoothing samples	Average filtering for display, reserved for legacy reasons, unused.	word
Monitor Mode	Monitor mode options: 0 = software mode, the voltage output on the MONITOR port is defined by the software value in this command 1 = high voltage monitor mode, the voltage on the MONITOR port is proportional to the output voltage, with 0 to +10 V corresponding to the 0 to maximum voltage range. 2 = position monitor mode, the voltage on the MONITOR port is proportional to the position, with 0 to +10 V corresponding to the 0 to maximum travel	word

	Note that in accordance with the fact that both the output voltage and position can be slightly negative, the voltage on the MONITOR output can also be slightly negative, typically down to -1 Volt. Any other value = reserved, unused	
Monitor Filter Frequency	Filter frequency for the MONITOR output, in Hz. If required, the MONITOR output can be filtered for increased noise reduction. A value of 0 turns filtering off.	word
Monitor software value	Software value for the MONITOR output when it is configured to software mode. Signed integer value of -3276 to 32767 corresponding to -1 V to +10 V.	
Reserved	Reserved, unused	word [3]

Example:

Rx: FC 07 20 00 D0 01 01 00 02 00 01 00 0C 00 01 00 00 20 00 00 00 40 00 00 64 00 01 00 14 00 32 73 00 00 00 00 00 00

Header: FC, 07, 20, 00, D0, 01 --> message ident 07FC followed by 32-byte data packet
Channel ident: 01 00 --> channel 1
Trigger I/O #1 Mode: 02 00 --> decimal 2, input trigger for voltage/position step up.
Trigger I/O #1 Polarity: 01 00 --> active high.
Trigger I/O #2 Mode: 0C 00 --> decimal 12, output active when strain gauge reading > lower limit.
Trigger I/O #2 Polarity: 01 00 --> active high.
Strain Gauge Lower Limit: 00 20 00 00 --> decimal 8192, equivalent to 25% of travel range
Strain Gauge Upper Limit: 00 40 00 00 -> decimal 16384, equivalent to 50% or travel range
Smoothing samples: 64 00 --> decimal 100
Monitor mode: 01 00 --> decimal 1, high voltage monitor mode.
Monitor Filter Frequency: 14 00 --> decimal 20, a 20 Hz cut-off frequency filter will be used.
Monitor Software Value: 32 73 --> decimal 29490, equivalent to 9 V.
Reserved: 00 00 00 00 00 00 --> unused bytes.

REQUEST:

Command structure (6 bytes)

0	1	2	3	4	5
Header					
FD	07	Chan ident	00	d	s

Channel Ident: reserved and ignored.

GET:

Response structure (38 bytes):
6 byte header followed by 32 bytes of data packet, as described for the SET command.

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
FE	07	20	00	d |0x80	s	Channel ident		Trig 1 Mode		Trig 2 Polarity	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Trig 2 Mode		Trig 2 Polarity		Strain Gauge Lower Limit				Strain Gauge Upper Limit			

24	25	26	27	28	29	30	31	32	33	34	35
Data											
SmSamples		Monitor Mode		Monitor Freq		Monitor Sw Val		Reserved		Reserved	

36	37
Data	
Reserved	

MGMSG_KPC_SET_KCUBEMMIPARAMS 0x08F0
MGMSG_KPC_REQ_KCUBEMMIPARAMS 0x08F1
MGMSG_KPC_GET_KCUBEMMIPARAMS 0x08F2

Function: It configures the operating parameters for the top panel LCD display and the associated control button and joystick wheel.

SET:
Command structure (40 bytes)
6 byte header followed by 32 bytes data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
F0	08	22	00	d |0x80	s	Channel ident		Joystick Mode		Joystick Gear	

12	13	14	15	16	17	18	19	20	21	22	23
Data											
Jstick Volt Step		Jstick Pos Step		Dir Sense		Preset Volt 1		Preset Volt 2		Preset Pos 1	

24	25	26	27	28	29	30	31	32	33	34	35
Data											
Preset Pos 2		Displ Bright		Displ Timeout		Displ Dim Level		Reserved		Reserved	

36	37	38	39
Reserved		Reserved	

Data Structure:

field	description	format
Channel indent	The channel being addressed	word
Joystick mode	This parameter specifies the operating mode of the joystick as follows: 1 = Voltage/Position Adjustment Mode: deflecting the joystick wheel changes the output voltage (in open loop mode) or position (in closed loop mode). The change is proportional to the deflection. The rate of change is set in the Joystick Gear parameter that follows. 2 = Jog Mode: deflecting the joystick wheel initiates a jog move, using the parameters specified by the Joystick Volt Step (in open loop) or Joystick Pos Step (in closed loop) parameter. One jog step per click of the wheel. 3 = Go To Mode: deflecting the joystick wheel starts a move from the current voltage or position to one of the two predefined “teach” voltages (in open loop) or positions (in closed loop). The teach voltages and positions are specified in the Preset Volt 1, Preset Volt 2, Preset Pos 1 and Preset Pos 2	word

	parameters. Any other value = reserved, unused	
Joystick Gear	The rate of change of voltage/position, when the JSMode parameter is set to Voltage/Position Adjust Mode. 1 = Voltage/position adjusts at a high rate, coarse adjustment mode 2 = Voltage/position adjusts at a medium rate 3 = Voltage/position adjusts at a low rate, fine adjustement mode Any other value = reserved, unused	word
Joystick Voltage Step	The voltage step size when Joystick mode is set to Jog Mode, with 0 to 32767 corresponding to the current maximum voltage	word
Joystick Position Step	The position step size when Joystick mode is set to Jog Mode, with 0 to 32767 corresponding to the current maximum travel range	word
Direction Sense	This parameter specifies the direction of a move initiated by the velocity wheel as follows: 0 = Wheel disabled. 1 = Normal: upwards rotation of the wheel results in an increased voltage/position. 2 = Reverse: upwards rotation of the wheel results in a decreased voltage/position. Any other value = reserved, unused	word
Preset Voltage 1	Preset voltage 1 when operating in Go to Mode, with 0 to 32767 corresponding to the current maximum voltage	word
Preset Voltage 2	Preset voltage 2 when operating in Go to Mode, with 0 to 32767 corresponding to the current maximum voltage	word
Preset Position 1	Preset position 1 when operating in Go to Mode, with 0 to 32767 corresponding to the maximum travel range	word
Preset Position 2	Preset voltage 2 when operating in Go to Mode, with 0 to 32767 corresponding to the maximum travel range	word
Display Brightness	Brightness level of the LCD display on the top of the unit. The brightness is set as a value from 0 (Off) to 100 (brightest). The display can be turned off completely by entering a setting of zero, however, pressing the MENU button on the top panel will temporarily illuminate the display at its lowest brightness setting to allow adjustments. When the display returns to its default position display mode, it will turn off again	word
Display Timeout	Timeout for the display dimming: the display is automatically dimmed after the time interval specified in the DispTimeout parameter has elapsed. Set in minutes in the range 0 (never dimmed) to 480.	word

	The dim level is set in the Display Dim Level parameter below	
Display Dim Level	The dim level, as a value from 0 (Off) to 10 (brightest) but is also limited by the DispBrightness parameter	word
Reserved	Reserved, unused	word [4]

Example:

Rx: F0 08 22 00 D0 01 01 00 01 00 03 00 66 06 66 06 01 00 CD 0C 99 19 CD 0C 99 19 32 00 00 00 19 00 00 00 00 00 00 00 00 00

Header: F0, 08, 22, 00, D0, 01 --> message ident 08F0 followed by 34-byte data packet.
Channel ident: 01 00 --> channel 1.
Joystick mode: 01 00 --> Voltage/Position Adjustment Mode.
Joystick Gear: 03 00 --> Voltage/position adjusts at a low rate, fine adjustment mode.
Joystick Voltage Step: 66 06 --> decimal 1638, equivalent to 5% of the maximum voltage.
Joystick Position Step: 66 06 --> decimal 1638, equivalent to 5% of the maximum travel range.
Direction Sense: 01 00 --> normal
Preset Voltage 1: CD 0C --> decimal 3277, 10% of the current maximum voltage.
Preset Voltage 2: 99 19 --> decimal 6553, 20% of the current maximum voltage.
Preset Position 1: CD 0C --> decimal 3277, 10% of the current maximum travel range.
Preset Position 2: 99 19 --> decimal 6553, 20% of the current maximum travel range.
Display Brightness: 32 00 --> decimal 50 (50%).
Display Timeout: 00 00 --> display is never dimmed.
Display Dim Level: 19 00 --> decimal 25 (25%).
Reserved: 00 00 00 00 00 00 00 00 --> unused bytes.

REQUEST:

Command structure (6 bytes)

0	1	2	3	4	5
Header					
F1	08	Chan ident	00	d	s

Channel Ident: reserved and ignored.

GET:

Response structure (40 bytes):
6 byte header followed by 34 bytes of data packet, as described for the SET command.

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
F2	08	22	00	d |0x80	s	Channel ident		Joystick Mode		Joystick Gear	

12	13	14	15	16	17	18	19	20	21	22	23

Data					
Jstick Volt Step	Jstick Pos Step	Dir Sense	Preset Volt 1	Preset Volt 2	Preset Pos 1

24	25	26	27	28	29	30	31	32	33	34	35
Data											
Preset Pos 2		Displ Bright		Displ Timeout		Displ Dim Level		Reserved		Reserved	

36	37	38	39
Reserved		Reserved	

MGMSG_KPC_SET_IOSETTINGS 0x08F3
MGMSG_KPC_REQ_IOSETTINGS 0x08F4
MGMSG_KPC_GET_IOSETTINGS 0x08F5

Function: It configures the analogue input and the maximum voltage range for the controller.
In the KPC101 controller the analogue input port can be configured to control voltage in open loop mode, position in closed loop mode, or used as a general-purpose analogue input with the voltage present on the input not affecting the operation of the piezo controller using the GMSG_PZ_SET_INPUTVOLTSSRC command.
In conjunction with this, the analogue input can be taken from the hub channels (hub channel A or B) or from the EXT IN SMA connector using this command. This command is also used to switch the strain gauge input between measuring position or measuring force.

SET:

Command structure (20 bytes)
6 byte header followed by 14 bytes data packet as follows:

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
F3	08	0E	00	d |0x80	s	Channel ident		Voltage Limit		Analog In Src	

12	13	14	15	16	17	18	19
Filter Freq		Voltage Range		Reserved		Reserved	

Data Structure:

field	description	format
Channel indent	The channel being addressed	word
Voltage Limit	Maximum output voltage in Volts	word
Analog Input Source	0=Disabled 1 = Hub input A 2 = Hub input B 3 = EXT IN connector Any other value = reserved, unused	word
Filter Frequency	Filter frequency in Hz. Filtering can be used to reduce noise on the input port at the expense of a slower response. Filtering can be turned off by setting this value to 0.	word
Voltage Range	Voltage range in Volts (75 or 150)	word
Force Sense	Maximum force corresponding to full scale reading in units of 10 mN.	word [2]

Options	Options for I/O settings: 0 = Strain gauge sensor reads position 1 = Strain gauge sensor reads force Any other value = reserved, do not use reading in units of 10 mN.	word [2]

Example:

Rx: F3 08 0E 00 D0 01 01 00 46 00 03 00 14 00 4B 00 08 0B 00 00

Header: F3, 08, 0E, 00, D0, 01 --> message ident 08F3 followed by 14-byte data packet.
Channel ident: 01 00 --> channel 1.
Voltage Limit: 46 00 --> 70 Volt maximum output voltage.
Analog Input Source: 03 00 --> EXT IN connector is used as an analogue input.
Filter Frequency: 14 00 --> decimal 20, corresponding to 20 Hz.
Voltage Range: 4B 00 --> 75 Volt hardware range.
Force Sense: 08 0B --> 3000 decimal, maximum force is 30 N.
Options: 00 00 --> Strain gauge input measures position.

Notes:

(a) The Voltage Range parameter must be one of two values for the KPC101 controller: 75 or 150.
(b) Voltage Limit must be a positive value less than or equal to Voltage Range.

REQUEST:

Command structure (6 bytes)

0	1	2	3	4	5
Header					
F4	08	Chan ident	00	d	s

Channel Ident: reserved and ignored.

GET:

Response structure (20 bytes):
6 byte header followed by 14 bytes of data packet, as described for the SET command.

0	1	2	3	4	5	6	7	8	9	10	11
Header						Data					
F5	08	0E	00	d |0x80	s	Channel ident		Voltage Limit		Analog In Src	

12	13	14	15	16	17	18	19
Filter Freq		Voltage Range		Reserved		Reserved	

Index