Kessler kOS
Second Shooter Plus Third Party
Network Protocol Definition

Last Updated: September 27, 2017

Introduction

This document describes the network communications between a kOS motor control product (Device) and client control applications. Messages between the Device and clients are categorized as Get/Set Messages, Action Messages, or Notification Messages. Each message consists of a fixed header along with a payload containing 0 or more parameter values. Messages have a message type in the header that indicates whether they are intended to 'Set' a value on the Device or 'Get' a value from the Device. All messages from the Device to clients have a message type of 'Response'.

Terminology

- "Device" refers to any kOS network-enabled motor control product, such as the CineDrive Brain or Second Shooter.
- A "move" refers to a series of pre-programmed motor moves (and/or photo shoots) that are executed by the Device.
- "Network" refers to any communication medium such as Ethernet or USB.
- The terms "client" and "software" are synonymous and refer to any implementation of kOS control software.
- A "message" is a discrete block of data sent over the network between a software client and a Device.
- The terms "motor" and "axis" are synonymous and refer to a single addressable motor.

Units

- Motor position, as well as the begin and end calibration points, are expressed in rotations.
- Motor speed is expressed in rotations per second.
- Motor acceleration/deceleration rates are expressed in rotations per second per second.
- Delay and duration values are expressed in seconds.
- When referring to a value within a specific range, Duration and Position can also be expressed as a 'normalized' value between 0 and 1.

Notes

- Many messages take a one byte motor address to deal with a specific motor. Motor addresses must be between 1 and 10, inclusive.
- In theory, a motor of address of 0 can be used to address all motors. In practice, this is only used in two cases: to reset all the axes at one time and when instructing an attached camera to take a picture.
- All data values are sent over the network in network (big endian) byte order.

Data Types

Parameter data types are specified as follows:

Data Type	Description
Bool	Unsigned 8-bit integer where 1 = True/On/Enabled and 0 = False/Off/Disabled
UInt8	Unsigned 8-bit integer.
UInt16	Unsigned 16-bit integer.
Int32	Signed 32-bit integer.
UInt32	Unsigned 32-bit integer.
Float32	32-bit floating point value.
UnixTime	Unsigned 32-bit integer specifying the number of seconds for a time interval or the number of seconds since 1-1-1970 12:00AM GMT for a date/time value.
String	Unicode (UTF-8) string value. The first two bytes are a UInt16 value specifying the length of the string in characters (not including the length bytes). Values may be null-terminated.
Block	Generic block of data. The first two bytes are UInt16 value specifying the length of the block in bytes (not including the length bytes).

Message Header

Messages contain both a header and a payload. The payload contains parameter values that are defined by the specific message, but every message has the following header:

Field	Data Type	Description
ProtocolVersion	UInt8	0x04
MessageLength	UInt16	The length of the message including the entire header.
MessageID	UInt16	The unique ID of the message.
MessageType	UInt8	The type of message: • 0 = Set • 1 = Get • 2 = Response (to a 'Get')

Get/Set Messages

- Clients always initiate Get/Set messages. These messages are used to share 'state' information between clients and the Device.
- Whether the message is a Get, Set, or a Response (to a Get) is determined by the MessageType field in the header.
- A message with type 'Response' should always be returned upon receiving a 'Get' message.
- No response is sent to a 'Set' message. If a Set will take a significant amount of time, or if the client needs to be informed when the Set is completed, an Action message with a Notification is a better implementation.
- Naming convention: The name of the message is descriptive of the data that is shared. Depending on the value of MessageType, a prefix of Get, a prefix of Set, or a suffix of Response can be added (e.g. GetMotorPosition, SetMotorPosition, MotorPositionResponse).

Msg ID	Message Name
0x02 (2)	NetworkInfo
0x07 (7)	LEDStatus
0x0B (11)	MotorInfo
0x0F (15)	DeviceInfo
0x10 (16)	UserPassword
0x11 (17)	DevicePassword
0x12 (18)	TrajectoryData
0x1F (31)	DeviceGUID
0x25 (37)	SlaveInfo
0x80 (128)	Action
0x81 (129)	Notification

NetworkInfo

Msg ID	Get Parameters	Set/Response Parameters
0x02 (2)	LimitedResponse (Bool)	DHCP (Bool) MACAddress (6 bytes) IPAddress (UInt32) SubnetMask (UInt32) Gateway (UInt32) DeviceType (UInt8) DeviceAddr (UInt8)

- This is used to get the current network settings of devices which communicate via Ethernet.
- This message is used for device discovery on the network. Ethernet devices should broadcast this message every five seconds when no clients have made a TCP connection.
- If LimitedResponse is true, the response is sent back using a UDP limited broadcast (255.255.255.255).
- The Gateway address is unused – discovery across routers is not supported.
- DeviceType is used to identify the type of kOS device. This should be set to 0 on a Set message and ignored by the receiving device.
- DeviceType currently defined values are as follows:
    - 0 = Unknown
    - 1 = CineDrive Brain
    - 2 = Second Shooter 1
    - 3 = Second Shooter 2
    - 128 = Client Software
- DeviceAddr is currently unused and should be set to 0.

LEDStatus

Msg ID	Get Parameters	Set/Response Parameters
0x07 (7)	(None required)	Master Status (Bool) Slave Status (Bool)

- Used to set or retrieve the state of the flashlight(s) on Second Shooter.
- The Status parameters can contain one of the following values:
    - 0 = Off
    - 1 = On
    - 2 = Slow Blink
    - 3 = Fast Blink

MotorInfo

Msg ID	Get Parameters	Response Parameters

0x0B (11)	(None required)	MotorCount (UInt8)
		  MotorAddress (UInt8)
		  Position (Float32)
		  EndPosition (Float32)
		  MaxMaxSetupSpeed (Float)
		  MaxMaxMoveSpeed (Float)
		  MaxMaxAcceleration (Float)
		(repeat for each MotorCount)

- GetMotorInfo can be used by clients to discover the motors that are connected to the Device. Subsequent changes to motor status are handled through the MotorStatus notification.
- SetMotorInfo is not supported.
- Only the first seven bits of MotorAddress should be used for the motor address. The high bit is set, to indicate that the motor is a F.I.Z. motor.
- An EndPosition of 0 indicates the axis has not been (or is no longer) calibrated.
- MaxMaxSetupSpeed is the maximum speed allowed by this motor when operating manually. It sets the upper bound when configuring 'Max Setup Speed' in kOS software.
- MaxMaxMoveSpeed is the maximum speed allowed by this motor when executing a move. It sets the upper bound when configuring 'Max Move Speed' in kOS software.
- MaxMaxAcceleration is the maximum acceleration allowed by this motor when executing a move. It sets the upper bound when configuring 'Damping' in kOS software.

DeviceInfo

Msg ID	Get Parameters	Response Parameters
0x0F (15)		DeviceType (UInt8) DeviceAddr ((Uint8) PlaybackMode (UInt8) PlaybackStatus (UInt8) FirmwareVersionMajor (UInt8) FirmwareVersionMinor (UInt8) FirmwareVersionRelease (UInt8) FirmwareVersionBuild (UInt8) NetworkID (UInt8) HardwareID (UInt8) DevicePassword (string) AuxInputStatus (UInt8) DelayTimeRemaining (Float32) ElapsedTime (Float32)

- SetDeviceInfo is not supported.
- Clients should send a GetDeviceInfo immediately after making a TCP connection.
- DeviceType is used to identify the type of Device (see the DeviceType parameter in the NetworkInfo message for the enumeration).
- DeviceAddr is currently unused and should be set to 0.
- NetworkID, HardwareID, and MotorProfileCount are not supported and should contain a value of 0.
- PlaybackMode indicates the last mode sent to the Device:
    - 0 = Live Motion
    - 1 = Time Lapse (Continuous)
    - 2 = Time Lapse (Shoot Move Shoot)
- PlaybackStatus indicates the status of a move when a move is in progress (see PlaybackStatus for possible values).
- AuxInputStatus: the status of the Aux input(s). See the AuxInputStatus notification message for possible values.
- If PlaybackStatus is DelayTimerStarted, then DelayTimeRemaining is the amount of time remaining (in seconds) before the move will begin.
- If PlaybackStatus indicates that a move is currently being executed, ElapsedTime is the time that has elapsed (expressed as the percentage of the total move time).

UserPassword

Msg ID	Get Parameters	Set Parameters
0x10 (16)	(Not supported)	Password (String)

- A SetUserPassword should be sent by a client to the device right after a TCP connection is made. The user password must match any password set previously on the device using the SetDevicePassword message.
- GetUserPassword is not needed in this scheme and is not supported.

DevicePassword

Msg ID	Get Parameters	Set Parameters
0x11 (17)	(Not supported)	Password (String)

- SetDevicePassword is used to set or change the Device password. Clients should follow this up with a SetUserPassword with the same value in order to maintain an unlocked state.
- GetDevicePassword is not supported. The Device password is included in the DeviceInfo payload.

TrajectoryData

Msg ID	Get Parameters	Set/Response Parameters

0x12 (18)	(None required)	BlockCount (UInt8)
		  | BlockID (UInt8)
		  | TrajectoryData (Block)
		(repeat for each BlockCount))

- TrajectoryData contains blocks of information required to execute a move on a Device (see the TrajectoryData section for the format of this data).
- A GetTrajectoryData message must generate a response, even if no TrajectoryData has previously been sent to the Device.

DeviceGUID

Msg ID	Get Parameters	Response Parameters
0x1F (31)	(None required)	GUID (Block)

- Used to retrieve the unique GUID for the device.
- SetDeviceGUI is not supported.

SlaveInfo

Msg ID	Get Parameters	Response Parameters
0x25 (37)		FirmwareVersionMajor (UInt8) FirmwareVersionMinor (UInt8) FirmwareVersionRelease (UInt8) FirmwareVersionBuild (UInt8)

- Used to retrieve the firmware version of a slave controller. If no slave controller is connected, '0.0.0.0' is returned.
- SetSlaveInfo is not supported.

Action Messages

- Action messages are always sent by clients to the Device. They are used to perform an action on the Device where the data involved does not need to be persistent. In several cases, the Device responds with a Notification when it completes the requested action.
- The MessageID for action messages is always 0x80.
- The MessageType is always 'Set'.
- The message payload consists of at least a single byte specifying the Action ID, but may also contain additional parameters depending on the action.
- Naming convention: verb describing the action followed by the target of that action.

Action ID	Action Name
0x01 (1)	ResetDevice
0x02 (2)	ResetAxis
0x03 (3)	MarkBeginPosition
0x04 (4)	MarkEndPosition
0x05 (5)	StartPlayback
0x06 (6)	StopPlayback
0x07 (7)	RapidToFirstKeyFrame
0x09 (9)	TakePicture
0x0F (15)	SetupPlayback
0x1C (28)	SetPositionSpeedAcceleration

ResetDevice

Action ID		Parameters
0x01 (1)		

- Performs a soft reboot of the Device.

ResetAxis

Action ID		Parameters
0x02 (2)		MotorAddress (UInt8)

- Resets an axis (i.e. clears the calibration).
- Clients can use an address of 0 to reset all axes.
- The current motor position is reset to 0 after this message is received by the device.

MarkBeginPosition

Action ID		Parameters
0x03 (3)		MotorAddress (UInt8)

- Sets the current motor position as the MarkBegin calibration point.

MarkEndPosition

Action ID		Parameters
0x04 (4)		MotorAddress (UInt8)

- Sets the MarkEnd calibration point.
- The Device sends a MotorCalibrated notification when calibration is completed. At that point, MarkBegin is 0, MarkEnd is a positive or negative value relative to MarkBegin, and the current motor position is equal to MarkEnd.

StartPlayback

Action ID		Parameters
0x05 (5)		Direction (UInt8) StartTime (Float32)

- Instructs the Device to start playback of a move (see the sections on Executing a Live Motion Move for details on how this is used).
- The Device responds with PlaybackStatus notifications.
- The Direction and StartTime parameters should be set to 0.
    - 0 = Forward
    - 1 = Reverse
- Note: when using firmware version 0.4.8.x or earlier, the device must have previously received (since the last network connection) a GetDeviceUUID message.

StopPlayback

Action ID		Parameters
0x06 (6)		

- Instructs the Device to stop a move and stop all motors immediately.
- The Device responds with PlaybackStatus notifications and MotorPosition notifications.

RapidToFirstKeyFrame

Action ID		Parameters
0x07 (7)		

- Instructs the Device to move all motors directly to their first key frame positions as quickly as possible.
- The Device performs this same function after receiving a SetupPlayback message (or a StartPlayback message not preceded by a SetupPlayback message).

TakePicture

Action ID		Parameters
0x09 (9)		MotorAddress (UInt8) Exposure (Float32)

- Instructs the Device to take a picture using an attached camera.
- Clients should always use a motor address of 0.
- Exposure is the time in seconds between opening and closing the shutter on the camera.

SetupPlayback

Action ID		Parameters
0x0F (15)		Direction (UInt8) StartTime (Float32) DelayReturn (Bool)

- Instructs the Device to completely load a move without actually starting the move (see the sections on Executing a Live Motion Move for details on how this is used).
- When this message is sent, the StartPlayback message is used to actually start the move.
- See StartPlayback for parameter definitions and restrictions.
- If DelayReturn is true, the return to first key frame does not happen until after the StartPlayback message is sent.
- The Device responds with PlaybackStatus notifications. A PlaybackStatus: SetupPlaybackCompleted notification is required.

SetPositionSpeedAcceleration

Action ID		Parameters
0x1C (28)		MotorAddress (UInt8) Position (Float32) Speed (Float32) Acceleration (Float32)

- Sets the position, speed and acceleration for the specified motor.
- On uncalibrated motors, Position should be +/-25000 (depending on what direction you want to move)
- On calibrated motors, Position should be between 0 and the Mark End position.
- The maximum allowable Speed is 165 for Slider/Pan, 65 for Tilt, and 200 for Focus/Zoom.
- Setting the motor speed to 0 stops the motor.
- The maximum allowable Acceleration is 100 for Slider/Pan and 50 for Tilt/Focus/Zoom.

Notification Messages

- Notification messages are always sent unsolicited by the Device to clients.
- The MessageID is always 0x81.
- MessageType is always 'Response'.
- Payload consists of at least a single byte specifying the Notification ID, but may also contain additional parameters depending on the notification.
- Naming convention: the name should be descriptive of the action or data that triggered the notification. The notification can also be referred to with Notification appended to the end (e.g. MotorStatusNotification).

Note ID	Notification Name
0x01 (1)	MotorStatus
0x02 (2)	PlaybackStatus
0x03 (3)	ErrorStatus
0x07 (7)	MotorPosition
0x11 (17)	MotorCalibrated
0x15 (21)	AuxInputStatus
0xFF (255)	UnsupportedMessage

MotorStatus

Note ID		Parameters

0x01 (1)		MotorCount (UInt8)
		  MotorAddress (UInt8)
		  OnlineStatus (Bool)
		  MotorType (UInt8)
		  MaxMaxSetupSpeed (Float)
		  MaxMaxMoveSpeed (Float)
		  MaxMaxAcceleration (Float)
		(repeat for each MotorCount)

- Sent every time the status of a motor changes (a motor comes online or goes offline).
- MotorType: 0 = (not F.I.Z), 1 = (F.I.Z)
- See the MotorInfo message for additional parameter information.

PlaybackStatus

Note ID		Parameters

0x02 (2)		PlaybackStatus (UInt8)
		ElapsedTime (Float32)
		MotorCount (UInt8)
		  MotorAddress (UInt8)
		  Position (Float32)
		(repeat for each MotorCount)

- Sent at key times during the preparation and execution of a move to indicate the current state.
- PlaybackStatus can be one of the following values:
    - 0 = NoMove: no move is being executed.
    - 1 = MotorsHome: motors have returned to their first key frame positions.
    - 2 = StartForward: a forward move has been started or is in progress.
    - 3 = StartReverse: a reverse move has been started or is in progress.
    - 4 = EndOfMove: the end or beginning of the move has been reached.
    - 5 = FullStop: the move has been stopped or has completed.
    - 6 = StartFrame: sent at the beginning of each a delay-exposure-move sequence in when executing a Time Lapse Shoot Move Shoot.
    - 7 = SetupCompleted: the Device has finished configuring the move after a SetupPlayback message has been received.
    - 8 = DelayTimerStarted: the Device has finished configuring the move and the delay timer has started.
    - 9 = PreCameraRollStarted: when the 'Camera Roll' feature is enabled, the Device has opened the shutter.
    - 10 = PostCameraRollCompleted: when the 'Camera Roll' feature is enabled, the Device has closed the shutter.
    - 11 = PreMovePhotosStarted: on a Time Lapse, the Device has started taking the pre-move photos.
    - 12 = PostMovePhotosCompleted: on a Time Lapse, the Device has completed taking the post-move photos.
    - 13 = SMSMotorsStopped: sent each time the motors stop during a Time Lapse Shoot Move Shoot.
- ElapsedTime is the normalized current elapsed time of the move.
- MotorCount/MotorAddress/Position: the current position of all the motors included in the move.

ErrorStatus

Note ID		Parameters
0x03 (3)		ErrorCode (UInt8) MotorAddress (UInt8)

- Used to notify the client of general error conditions.
- ErrorCode is defined as follows:
    - 16 = A fault condition was detected - axis will be reset
- MotorAddress is the address of the motor that triggered the error.

MotorPosition

Note ID		Parameters
0x07 (7)		MotorAddress (UInt8) Position (Float32)

- Sent after a motor has become idle after the motor's position has been changed.
- *This notification should be sent after any attempt by client software to change the motor position, even if the motor does not actually change position.*

MotorCalibrated

Note ID		Parameters
0x11 (17)		MotorAddress (UInt8) EndPosition (Float32)

- Sent by the Device when a motor has been successfully calibrated. This is the expected response to a MarkEndPosition action message.

AuxInputStatus

Note ID		Parameters
0x15 (21)		Status (UInt8)

- Sent by Second Shooter when the status of one of the aux inputs changes or when a slave controller is connected or disconnected.
- The Status byte is a bitmapped value and should be interpreted as follows:
  - Bit 0: Master tip
  - Bit 1: Master ring
  - Bit 2: Slave tip
  - Bit 3: Slave ring
  - Bit 4: Slave controller is present

UnsupportedMessage

Note ID		Parameters
0xFF (255)		

- Sent by Second Shooter when it receives a message it does not support. The payload contains as much of the offending message as possible.

Initialization

The following table describes initialization sequence takes place whenever Kessler kOS software attempts to establish a network connection to the Device. This happens when the software is first started, when the software becomes active after sleeping, or whenever the software attempts to reestablish a connection. The latter can be user initiated or can happen, for example, after the Device has been reset, or after new firmware has been uploaded.

Device Discovery

The Device should send a NetworkInfo response as a limited Broadcast every five seconds when a TCP connection has not been made to the Device.

Initialization

The following assumes that the Client has made a TCP connection to the Device.

Client	Device
Send GetDeviceInfo	
	Send DeviceInfoResponse
[1]Receive DeviceInfoResponse [2]Send SetUserPassword Send GetMotorInfo	
	Send MotorInfoResponse
[3]Receive MotorInfoResponse [4]Send GetDeviceGUID	

[1]The software GUI is disabled after a TCP connection is made. There is a ten second timeout when waiting to receive a DeviceInfoResponse.
[2]kOS sends this message if a password has been configured on the device. It is not required otherwise.
[3] The software GUI is enabled after receiving the MotorInfoResponse or after the timeout expires.
[4]Firmware v0.4.8.x or earlier: the device must receive this message before it will respond to the StartPlayback message.

TrajectoryData

Format

BlockCount (Uint8)

Playback (BlockID=1)
BlockID (Uint8) = 1
BlockSize (Uint16) = 33
PlaybackMode (UInt8)
Duration (Float32)
RapidReverseDuration (Float32)
LoopAfterRapidReverse (UInt8)
LoopAfterReverse (UInt8)
DelayMode (UInt8)
DelayInterval (Float32)
CameraRollEnabled (Bool)
CameraRollPreDelay (Float32)
CameraRollPostDelay (Float32)
FlashMarkEnabled (Bool)
FlashMarkType (UInt8)
FlashMarkPostDelay (Float32)
UseExternalTrigger (Bool)
UsingLANCCable (Bool)

TimeLapse (BlockID=2)
BlockID (Uint8) = 2
BlockSize (Uint16) = 21
Delay (Float32)
Exposure (Float32)
MovePhotos (UInt32)
Not Used (UInt8)
PreMovePhotos (UInt32)
PostMovePhotos (UInt32)

StopMotion (BlockID=3)
BlockID (Uint8) = 3
BlockSize (Uint16) = 10
Exposure (Float32)
Photos (UInt32)
AutoAdvance (Bool)
Not Used (UInt8)

Axis (BlockID=5) (One for each axis included in the move)
BlockID (Uint8) = 5
BlockSize (Uint16) = 2
MotorAddress (UInt8)
SegmentCount (UInt8)

Segment (BlockID=6) (One for each segment of each axis included in the move)
BlockID (Uint8) = 6
BlockSize (Uint16) = 34
MotorAddress (UInt8)
SegmentNumber (UInt8) (0-based)
P0Time (Float32)
P0Position (Float32)
P1Time (Float32)
P1Position (Float32)
P2Time (Float32)
P2Position (Float32)
P3Time (Float32)
P3Position (Float32)

Notes

- The first three blocks must be included in the payload of every SetTrajectoryData message sent.
- Include one Axis block (and the desired number of Segment blocks) for each axis included in the move.
- Fields labeled '*Not Used*' must be included and set to 0.

Playback

- PlaybackMode – see DeviceInfo for possible values.
- Duration indicates the length of the actual move in seconds. It does not include pre/post move delays, or the length of time it takes to execute pre/post move photos. Note that Duration for a time lapse is not used by the Device as the length of the move is calculated based on delay, exposure, and the number of photos.
- RapidReverseDuration – the move time to use when moving when performing a rapid reverse while looping.
  This value should be set to the same value as Duration.
- LoopAfterReverse: indicates the number of times to loop after executing the move in reverse. A value of 255 indicates the loop should be executed continuously.
- If LoopAfterReverse is non-zero, when the move gets to time 1.0, the motors are returned to their first key frame positions by following the trajectory. Duration is used as the length of the move in reverse. Once the motors reach their first key frame positions, the move is executed again.
- LoopAfterRapidReverse: indicates the number of times to loop after executing the move rapidly in reverse. A value of 255 indicates the loop should be executed continuously.
- If LoopAfterRapidReverse is non-zero, when the move gets to time 1.0, the motors are returned to their first key frame positions by following the trajectory. RapidReverseDuration is used as the length of the move in reverse. Once the motors reach their first key frame positions, the move is executed again.
- LoopAfterReverse and LoopAfterRapidReverse are mutually exclusive. Only one should be set to a non-zero value.
- Looping is not supported when performing a Time Lapse and these values should be set to 0.
- DelayMode:
  - 0 = No delay.
  - 1 = Delay for time (DelayInterval is the number of seconds to delay).
  - 2 = Delay until a specific date/time (DelayInterval is the date/time to start the move in UnixTime format).
- Camera Roll is not supported when performing a Time Lapse and CameraRollEnabled should be set to 0.
- CameraRollPreDelay should be set to 0. Use the DelayInterval field to specify the length of time to delay before executing the move.
- All FlashMark fields should be set to 0.
- UseExternalTrigger and UsingLANCCable are unsupported and should be set to 0.

Time Lapse

- Delay and Exposure are in seconds.

StopMotion

- Exposure, Photos, and AutoAdvance should all be set to 0.

Axis/Segment

- For each segment, P0 and P3 are the end points and P1 and P2 are the Bezier control points between the end points. Both time and position are normalized values. Time is expressed as a percentage of the Duration of the move and Position is expressed as a percentage of the calibrated range for the motor.

Executing a Live Motion Move

The following table lists the actions and notifications during the loading and executing of a Live Motion move:

Action	Client	Device
User selects 'Load'	Send SetTrajectoryData Send SetupPlayback	
[1]Device moves all motors to their first key frame position		Send PlaybackStatus: MotorsHome
Device has configured move and is ready to execute		Send PlaybackStatus: SetupCompleted
[2]User selects 'Play'	Send StartPlayback	
'Camera Roll' is enabled		Open camera shutter Send PlaybackStatus: PreCameraRollStarted
[3]The move has been configured with any delay and the device has started its internal countdown		Send PlaybackStatus: DelayTimerStarted
Delay countdown has expired and the device begins executing the move		Send PlaybackStatus: StartForward (or StartReverse)
The move reaches time 1.0 (or time 0.0 on a reverse move)		Send PlaybackStatus: EndOfMove
Looping is enabled		Repeat previous two steps as required
'Camera Roll' is enabled		Wait for any PostCameraRollDelay Close camera shutter Send PlaybackStatus: PostCameraRollCompleted
Move has been fully executed		[4]Send PlaybackStatus: FullStop
User selects 'Stop' at any time	Send StopPlayback	[4]Send PlaybackStatus: FullStop Send MotorPosition notification for all motors

Notes:

Looping is disabled if Camera Trigger is enabled.

[1]The SetupPlayback message has an option to delay this step until immediately after StartPlayback has been sent.

[2]It is also possible to forgo the SetupPlayback message and just send a StartPlayback message. In this case, all configuration is done on the device before subsequent steps are taken (a SetupCompleted notification is not sent).

[3] If Delay is enabled, DelayInterval field in the TrajectoryData will contains the highest value entered by the user for Delay, Camera Roll Pre Roll Delay, or Flash Mark Delay.

[4]When the Device sends "FullStop' the configured move is no longer valid (SetTrajectoryData must be sent again).

Executing a Time Lapse Move

The following table lists the actions and notifications during the loading and executing of a Time Lapse move:

Action	Client	Device
User selects 'Load'	Send SetTrajectoryData Send SetupPlayback	
[1]Device moves all motors to their first key frame position		Send PlaybackStatus: MotorsHome
Device has configured move and is ready to execute		Send PlaybackStatus: SetupCompleted
[2]User selects 'Play'	Send StartPlayback	
[3]The move has been configured with any delay and the device has started its internal countdown		Send PlaybackStatus: DelayTimerStarted
Begin executing pre move photos		Send PlaybackStatus: PreMovePhotosStarted
The device begins executing the move		Send PlaybackStatus: StartForward
Shoot-Move-Shoot only: At the beginning of each delay-picture-exposure sequence		Send PlaybackStatus: StartFrame
The move reaches time 1.0		Send PlaybackStatus: EndOfMove
Begin executing post move photos		Send PlaybackStatus: PostMovePhotosStarted
Move has been fully executed		[4]Send PlaybackStatus: FullStop
User selects 'Stop' at any time	Send StopPlayback	[4]Send PlaybackStatus: FullStop Send MotorPosition notification for all motors

Notes:

Looping and Camera Roll are disabled when executing a Time Lapse.
Reverse moves are not allowed when executing a Time Lapse.

[1]The SetupPlayback message has an option to delay this step until immediately after StartPlayback has been sent.

[2]It is also possible to forgo the SetupPlayback message and just send a StartPlayback message. In this case, all configuration is done on the device before subsequent steps are taken (a SetupCompleted notification is not sent).

[3]If Delay is enabled, DelayInterval field in the TrajectoryData will contains the highest value entered by the user for Delay or Flash Mark Post Delay.

[4]When the Device sends "FullStop' the configured move is no longer valid (SetTrajectoryData must be sent again).

3rd Party Development Notes

- Make sure you can connect to the Device from one of the kOS client applications (preferably on the computer you are using for development), then retrieve the IP address of the Device from the 'About' window of that application.
- Close any kOS client applications and disconnect all computers from the Device's wireless network except the one you are using for development.
- Establish a TCP connection to the Device (on port 5520) and use that connection to send all messages.
- There is a 30 second keep alive period on the TCP connection. You must send some message to the Device within that timeframe or the TCP connection will be dropped by the Device. If the client is idle, a GetNetworkInfo message should be sent every ten seconds to keep the connection alive.
- After a TCP connection has been made, send a GetDeviceInfo message. The Device will respond with a DeviceInfoResponse message.
- After exchanging the DeviceInfo messages, a SetUserPassword message containing the current Device password must be sent before any other messages can be sent.
- Use the GetMotorInfo message to retrieve information on the motors connected to the Device.
- To move uncalibrated motors, use the SetPositionSpeedAcceleration message:
    - The Position parameter should be either -25000 or 25000 (depending on which direction you want to go).
    - See the message documentation for maximum values for speed and acceleration.
    - Set the Speed parameter to 0 to stop the motor.
- To calibrate a motor:
    - Move the motor to the desired start position then send a MarkBeginPosition message.
    - Move the motor to the desired end position then send a MarkEndPosition message.
    - Wait for the MotorsCalibrated notification and note the Mark End position in the payload (this can be either a positive or negative number). At this point, the Mark Begin position is 0, and the Mark End position is a value relative to the Mark Begin position.
- To move calibrated motors, use the SetPositionSpeedAcceleration message:
    - The Position parameter should be between 0 and the Mark End position.
    - See the message documentation for maximum values for speed and acceleration.
- To execute a move, refer to the steps under Executing a Live Motion Move or Executing a Time Lapse Move above.

Discovery and Example Messages (Second Shooter Plus)

Make a TCP connection to Second Shooter (SS)

SS sends a NetworkInfo response as a UDP broadcast every five seconds that contains its IP address. You can also obtain the IP address from the About window in the kOS software.

Get Information about the Device

Immediately after making a TCP connection to the device, send a (Get)DeviceInfo message:

```
04 00 06 00 0F 01
```

You will receive a DeviceInfo response.

Get Information about attached motors

Send a (Get)MotorInfo message:

```
04 00 06 00 0B 01
```

You will receive a MotorInfo response for each motor attached to the device:

An example response:

```
04      //Protocol ID always 4
00 1C   //Message Size
00 0B   //Message ID
82      //Message Type: normally this would be 02 for a Response (see below)
01      //Number of motors in this message (see below)
01      //The motor address for this motor
C2 41 70 00    //The current position = 49.35938
C3 54 20 40    //The calibrated end position for this motor = 212.126 (see below)
43 26 B3 33    //The fastest speed for this motor during setup = 10002
43 26 B3 33    //The fastest speed for this motor in a programmed move = 10002
43 48 00 00    //The fastest acceleration for this motor = 200
```

Number of motors: you will receive one MotorInfo response per motor attached to SS, so this value should always be 01.
Message Type: if the value is 82 instead of 02, it indicates that more MotorInfo responses follow this one. The last response will have a Message Type of 02.
End Position: if this value is 0, it indicates the motor is not calibrated. If it is not 0, then the motor position can be between 0 and this number (which can be negative).

Moving a calibrated motor always involves setting an actual motor position (as opposed to a relative position), so you have to keep track of the current position of the motor at all times.

Move an uncalibrated motor

Use the SetMotorPositionSpeedAcceleration message. Set the motor position to either 25000 or -25000 (depending on which direction you want to move). Set the speed and acceleration as desired. *Set the motor speed to 0 to stop the motor.*

Example:

Send SetMotorPositionSpeedAcceleration: motorAddr=3 position=25000.000 speed=2.000 acceleration=44.000

```
04 00 14 00 80 00   1C   03   46 C3 50 00   3F FF FF F0   42 30 00 00
```

Move a calibrated motor

Use the SetMotorPositionSpeedAcceleration message. Remember that the position has to be between 0 and the calibrated end position returned by MotorInfo.

Example:

Send SetMotorPositionSpeedAcceleration motorAddr=1 position=50.910 speed=100.000 acceleration=100.000

```
04 00 10 00 80 00   1C   01    C2 4B A4 14    42 C8 00 00   42 C8 00 00
```

When the motors become idle…

You will receive a MotorPosition notification for each motor with the current position of the motor:

Example:

MotorPositionNotification: motorAddr=3 position=11.982

```
04 00 0C 00 81 02 07   03   41 3F B8 00
```