MultiClamp Commander Telegraph Interface Specification

Version 2.0, revised Sept 23, 2003

<u>Revision Notes for Version 2.0 (MultiClamp 700B)</u>

MultiClamp Commander 2.0 supports a new telegraph packet format for the MultiClamp 700B based on the device serial number and channel ID. The telegraph format for MultiClamp Commander 1.1 and MultiClamp 700A based on COM port, AxoBus ID and Channel ID is supported in parallel with this new format. The telegraph packet struct MC_TELEGRAPH_DATA supports both formats.

The telegraph API version MCTG_API_VERSION has been bumped to 12.

The following data has been added to the MC_TELEGRAPH_DATA data struct:

• dSecondaryAlpha	Output gain for secondary output signal.
• dSecondaryLPFCutoff	Low pass filter cutoff for secondary output signal.
• szAppVersion[16]	Application version number
• szFirmwareVersion[16]	Firmware version number
• szDSPVersion[16]	DSP version number
• szSerialNumber[16]	Serial number of device

<u>Revision Notes for Version 1.1 (MultiClamp 700A)</u>

Axon has revised its naming convention for VClamp output signals to more accurately reflect the nature of the signal:

- Command Potential has been renamed Membrane Potential. Membrane Potential does not include a Pipette Offset unless further specified by the signal name.
- Membrane Potential has been renamed Pipette Potential. Pipette Potential includes any Pipette Offset.

The following data has been added to the MC_TELEGRAPH_DATA data struct:

• uRawOutSignal	Raw or secondary output signal
• dRawScaleFactor	Raw or secondary output scale factor
• uRawScaleFactorUnits	Raw or secondary output signal
• uHardwareType	identifies MultiClamp model

The following registered messages have been added to the SDK:

```
static const char* MCTG_BROADCAST_MESSAGE_STR  = "MultiClampTelegraphBroadcastMsg";
static const char* MCTG_ID_MESSAGE_STR         = "MultiClampTelegraphIdMsg";
```

Further information on these changes can be found in this document.

MultiClamp Commander Telegraph Software Development Kit (SDK)

This document is part of the MultiClamp Commander Telegraph SDK which consists of the following parts:

- **MultiClamp Commander Telegraph Interface Specification (version 2.0) (this document)**
- **MultiClamp Commander Telegraph Interface Header File - MCTelegraphs.hpp (version 11)**
- **MFC Reference Project – MCTeleClient (version 2.0)**

MultiClamp Commander Telegraph Data

The MultiClamp Commander software can provide information about the state of the MultiClamp system to other Windows applications by way of a registered messaging interface. These MultiClamp data messages are called telegraph packets. The tables below specify the MultiClamp parameters that are contained in a telegraph packet.

Parameter	uOperatingMode
Description	Operating mode of amplifier
Type	UINT
Units	N/A
Values	V-Clamp I-Clamp I = 0

Parameter	uScaledOutSignal
Description	Signal identifier of scaled output (700A) or primary output (700B)
Type	UINT
Units	N/A
Values (700A)	Membrane Potential / Command Current Membrane Current Pipette Potential/ Membrane Potential x100 AC Pipette/ Membrane Potential Bath Current Bath Potential
Values (700B)	See "700B Primary and Secondary Output Signal Constants" section in MCTelegrpahs.hpp

Parameter	dAlpha
Description	Gain of scaled output (700B) or primary output (700B)
Type	Double
Units	Dimensionless

Parameter	dScaleFactor
Description	Scale factor of scaled output (700A) or primary output (700B)
Type	Double
Units	Variable

Parameter	uScaleFactorUnits
Description	Scale factor units of scaled output (700A) or primary output (700B)
Type	UINT
Units	Variable
Values	V / V V / mV V / µV V / A V / mA V / µA V / nA V / pA

Parameter	dLPFCutoff
Description	Lowpass filter cutoff of scaled output (700A) or primary output (700B)
Type	double
Units	Hertz

Parameter	dMembraneCap
Description	Membrane capacitance
Type	double
Units	Farads

Parameter	dExtCmdSens
Description	External command sensitivity
Type	double
Units	V/V *(V-Clamp)* A/V *(I-Clamp)* 0.0 A/V, OFF *(I=0)*

Parameter	uRawOutSignal
Description	Signal identifier of raw output (700A) or secondary output (700B)
Type	UINT
Units	N/A
Values (700A)	Membrane plus Offset Potential/ Command Current Membrane Current Pipette Potential/ Membrane plus Offset Potential x100 AC Pipette/ Membrane Potential Bath Current Bath Potential
Values (700B)	See "700B Primary and Secondary Output Signal Constants" section in MCTelegrpahs.hpp

Parameter	dRawScaleFactor
Description	Gain scale factor of raw output (700A) or secondary output (700B)
Type	Double
Units	Variable

Parameter	uRawScaleFactorUnits
Description	Gain scale factor units of raw output (700A) or secondary output (700B)
Type	UINT
Units	Variable
Values	V / V V / mV V / µV V / A V / mA V / µA V / nA V / pA

Parameter	uHardwareType
Description	Hardware type identifier
Type	UINT
Units	Dimensionless
Values	MCTG_HW_TYPE_MC700A MCTG_HW_TYPE_MC700B

Parameter	dSecondaryAlpha
Description	Gain of raw output (700A) or secondary output (700B)
Type	Double
Units	Dimensionless

Parameter	dSecondaryLPFCutoff
Description	Lowpass filter cutoff of raw output (700A) or secondary output (700B)
Type	double
Units	Hertz

Parameter	szAppVersion
Description	Application version of MultiClamp Commander 2.x
Type	char[16]

Parameter	szFirmwareVersion
Description	Firmware version of MultiClamp 700B
Type	char[16]

Parameter	szDSPVersion
Description	DSP version of MultiClamp 700B
Type	char[16]

Parameter	szSerialNumber
Description	Serial number of MultiClamp 700B
Type	char[16]

MultiClamp Commander 1.1 Telegraph Signal Identifiers

A MultiClamp telegraph connection must uniquely identify the amplifier channel to which the telegraph data belongs. The following parameters (signal identifiers) are required to identify a specific MultiClamp 700A channel on the computer system:

- **COM Port ID**
- **AxoBus ID (A.K.A. Device Number)**
- **Amplifier Channel ID**

The MultiClamp Commander telegraph messages pass the signal IDs in the form of a 4-byte bitfield with the following structure:

Byte 3	*Byte 2*	*Byte 1*	*Byte 0*
Channel ID (High)	**Channel ID (Low)**	**AxoBus ID**	**COM Port ID**

MCTelegraphs.hpp provides a couple of inline utility functions for handling this packed data format for the signal IDs.

```
inline LPARAM MCTG_Pack700ASignalIDs( UINT uComPortID,
                                      UINT uAxoBusID,
                                      UINT uChannelID  )

inline BOOL MCTG_Unpack700ASignalIDs( LPARAM lparamSignalIDs,
                                      UINT *puComPortID,
                                      UINT *puAxoBusID,
                                      UINT *puChannelID       )

inline BOOL   MCTG_Match700ASignalIDs( UINT   uComPortID,
                                       UINT   uAxoBusID,
                                       UINT   uChannelID,
                                       LPARAM lparamSignalIDs )
```

MultiClamp Commander 2.0 Telegraph Signal Identifiers

A MultiClamp telegraph connection must uniquely identify the amplifier channel to which the telegraph data belongs. The following parameters (signal identifiers) are required to identify a specific MultiClamp 700B channel on the computer system:

- **Serial Number**
- **Amplifier Channel ID**

The MultiClamp Commander telegraph messages pass the signal IDs in the form of a 4-byte bitfield with the following structure:

Byte 3 (high nibble)	*Byte 3 (low nibble)*	*Byte 2*	*Byte 1*	*Byte 0*
Channel ID (4 bits)	**Serial Number (28 bits)**			

MCTelegraphs.hpp provides a couple of inline utility functions for handling this packed data format for the signal IDs.

```
inline LPARAM MCTG_Pack700BSignalIDs( UINT uSerialNum,
                                      UINT uChannelID  )

inline BOOL MCTG_Unpack700BSignalIDs( LPARAM lparamSignalIDs,
                                      UINT   *puSerialNum,
                                      UINT   *puChannelID       )

inline BOOL   MCTG_Match700BSignalIDs( UINT   uSerialNum,
                                       UINT   uChannelID,
                                              LPARAM lparamSignalIDs )
```

MultiClamp Commander Telegraph Client Messages

The tables below summarize the Windows messages which are sent by a client application to communicate with the MultiClamp commander.

MessageID	RegisterWindowMessage(MCTG_OPEN_MESSAGE_STR)
Sender	Client
Purpose	Opens a telegraph connection to the MultiClamp Commander
wParam	Client hWnd
lParam	4-byte bitfield containing: 700A format (COM Port ID, AxoBus ID, Channel ID) 700B format (Serial Number, Channel ID)

MessageID	RegisterWindowMessage(MCTG_CLOSE_MESSAGE_STR)
Sender	Client
Purpose	Closes a telegraph connection to the MultiClamp Commander
wParam	Client hWnd
lParam	4-byte bitfield containing: 700A format (COM Port ID, AxoBus ID, Channel ID) 700B format (Serial Number, Channel ID)

MessageID	RegisterWindowMessage(MCTG_REQUEST_MESSAGE_STR)
Sender	Client
Purpose	Requests a telegraph packet from the MultiClamp Commander
wParam	Client hWnd
lParam	700A format (COM Port ID, AxoBus ID, Channel ID) 700B format (Serial Number, Channel ID)

MessageID	RegisterWindowMessage(MCTG_BROADCAST_MESSAGE_STR)
Sender	Client
Purpose	Broadcast to all MultiClamp Commanders to identify themselves with an MCTG_ID_MESSAGE_STR response.
wParam	Client hWnd
lParam	Not used

MultiClamp Commander Telegraph Server Messages

The tables below summarize the Windows messages which are sent by the MultiClamp Commander to communicate with a telegraph client application.

MessageID	RegisterWindowMessage(MCTG_RECONNECT_MESSAGE_STR)
Sender	Server (MultiClamp Commander)
Purpose	Requests the client to reopen an existing telegraph connection to the MultiClamp Commander
wParam	Not used
lParam	4-byte bitfield containing: 700A format (COM Port ID, AxoBus ID, Channel ID) 700B format (Serial Number, Channel ID)

MessageID	RegisterWindowMessage(MCTG_ID_MESSAGE_STR)
Sender	Server (MultiClamp Commander)
Purpose	Every instance of MultiClamp Commander will respond with this message upon receiving an MCTG_BROADCAST_MESSAGE_STR
wParam	Not used
lParam	4-byte bitfield containing: 700A format (COM Port ID, AxoBus ID, Channel ID) 700B format (Serial Number, Channel ID)

MessageID	WM_COPYDATA
Sender	Server (MultiClamp Commander)
Purpose	This is a standard Windows message for passing application-specific data. The MultiClamp Commander uses it to send telegraph packets to its clients.
wParam	Not used
lParam	pointer to COPYDATASTRUCT with telegraph data

COPYDATASTRUCT is a standard Windows structure. In application to MultiClamp Commander telegraphing, the members are to be interpreted as follows:

```
typedef struct tagCOPYDATASTRUCT
{
   DWORD   dwData;    // RegisterWindowMessage( MCTG_REQUEST_MESSAGE_STR )
   DWORD   cbData;    // size (in bytes) of the MC_TELEGRAPH_DATA structure being sent
   PVOID   lpData;    // pointer to the MC_TELEGRAPH_DATA structure
} COPYDATASTRUCT;
```

The MC_TELEGRAPH_DATA structure is defined and documented in MCTelegraphs.hpp.

Message Registration

MCTelegraphs.hpp defines the MultiClamp Commander telegraph interface registered messages as string constants. Before using these messages, they must first be registered to obtain the corresponding integer identifiers.

```
static UINT s_uMCTGOpenMessage      = 0;
static UINT s_uMCTGCloseMessage     = 0;
static UINT s_uMCTGRequestMessage   = 0;
static UINT s_uMCTGReconnectMessage = 0;
static UINT s_uMCTGBroadcastMessage = 0;
static UINT s_uMCTGIDMessage = 0;

s_uMCTGOpenMessage = RegisterWindowMessage( MCTG_OPEN_MESSAGE_STR );
ASSERT( s_uMCTGOpenMessage != 0 );

s_uMCTGCloseMessage = RegisterWindowMessage( MCTG_CLOSE_MESSAGE_STR );
ASSERT( s_uMCTGCloseMessage != 0 );

s_uMCTGRequestMessage = RegisterWindowMessage( MCTG_REQUEST_MESSAGE_STR );
ASSERT( s_uMCTGRequestMessage != 0 );

s_uMCTGReconnectMessage = RegisterWindowMessage( MCTG_RECONNECT_MESSAGE_STR );
ASSERT( s_uMCTGReconnectMessage != 0 );

s_uMCTGBroadcastMessage = RegisterWindowMessage( MCTG_BROADCAST_MESSAGE_STR );
ASSERT(s_uMCTGBroadcastMessage != 0 );

s_uMCTGIDMessage = RegisterWindowMessage( MCTG_ID_MESSAGE_STR );
ASSERT(s_uMCTGIDMessage != 0 );
```

These registered message identifiers can be used in the same way as the standard system-defined WM_XXX IDs. They are guaranteed to be unique within the system (at the time they are registered).

Opening a Telegraph Connection

Construct an LPARAM (4 byte) bitfield containing the desired signal identifiers.

```
// MultiClamp Commander 1.1 (700A)
LPARAM lparamSignalIDs = MCTG_Pack700ASignalIDs( uComPortID,
                                                 uAxoBusID,
                                                 uChannelID  );
// MultiClamp Commander 2.0 (700B)
LPARAM lparamSignalIDs = MCTG_Pack700BSignalIDs( uSerialNum,
                                                 uChannelID  );
```

Broadcast the telegraph open message to all top-level windows.

```
// Use your client's HWND as the wParam.
if( !::PostMessage( HWND_BROADCAST,
                    s_uMCTGOpenMessage,
                    (WPARAM) m_hWnd,
                    lparamSignalIDs        ) )
{
   AfxMessageBox("Failed to request open connection!");
}
```

If there is a MultiClamp telegraph server available to provide information about the specified device / channel, you will receive a WM_COPYDATA message containing an MC_TELEGRAPH_DATA structure for the MultiClamp device / channel you have specified. (see "Message Screening for WM_COPYDATA" below) If you do not receive this message shortly (within say, 1 sec) you should assume that there is no telegraph server for your request.

Once you have connected, you will continue to receive these messages whenever the server has changes to report.

Handling connections with multiple servers.

If you would like to connect to one or more servers when multiple instances of MultiClamp Commander are running, you may broadcast to all servers to identify themselves, and they will each respond.

Broadcast the telegraph broadcast message to all top-level windows.

```
//   Use your client's HWND as the wParam.

if( !::PostMessage( HWND_BROADCAST,
                    s_uMCTGBroadcastMessage,
                    (WPARAM) m_hWnd,
                    (LPARAM) 0        ) )
{
   AfxMessageBox("Failed to broadcast to telegraph servers!");
}
```

Every MultiClamp telegraph server available will provide information about the specified device / channel. You will receive a telegraph id message from each server.

```
// MultiClamp Commander 1.1 (700A)

// process telegraph id message
if( message == s_uMCTGIDMessage )
{
   // display the identification details of the server.
   UINT uComPortID = 0;
   UINT uAxoBusID = 0;
   UINT uChannelID = 0;
   if( !MCTG_Unpack700ASignalIDs( lParam, &uComPortID, &uAxoBusID, &uChannelID ) )
   {
      return TRUE; // Message was handled.
   }

   // Do something here with your new server information…

   return TRUE;
}

// MultiClamp Commander 2.0 (700B)

// process telegraph id message
if( message == s_uMCTGIDMessage )
{
   // display the identification details of the server.
   UINT uSerialNum = 0;
   UINT uChannelID = 0;
   if( !MCTG_Unpack700BSignalIDs( lParam, &uSerialNum, &uChannelID ) )
   {
      return TRUE; // Message was handled.
   }

   // Do something here with your new server information…

   return TRUE;
}
```

Closing a Telegraph Connection

When you are finished with the telegraph server, close your connection to make it available to other clients.

```
// Broadcast the telegraph close message to all top level windows.

// MultiClamp Commander 1.1 (700A)
```

```
LPARAM lparamSignalIDs = MCTG_Pack700ASignalIDs( uComPortID,
                                                 uAxoBusID,
                                                 uChannelID  );
// MultiClamp Commander 2.0 (700B)
LPARAM lparamSignalIDs = MCTG_Pack700BSignalIDs( uSerialNum,
                                                 uChannelID  );

// Use your client's HWND as the wParam.

if( !::PostMessage( HWND_BROADCAST,
                    s_uMCTGCloseMessage,
                    (WPARAM) m_hWnd,
                    lparamSignalIDs      ) )
{
   AfxMessageBox("Failed to request close connection!");
}
```

Requesting a Telegraph Packet

The MultiClamp Commander will push telegraph packets out to all subscribers (i.e. clients who have opened a connection) whenever there are changes to the relevant data.   However, there is also a request mechanism available for clients that do not wish to subscribe and want to receive telegraph packets only when they request them.

```
// Broadcast the telegraph request message to all top level windows.

// MultiClamp Commander 1.1 (700A)
LPARAM lparamSignalIDs = MCTG_Pack700ASignalIDs( uComPortID,
                                                 uAxoBusID,
                                                 uChannelID  );

// MultiClamp Commander 2.0 (700B)
LPARAM lparamSignalIDs = MCTG_Pack700BSignalIDs( uSerialNum,
                                                 uChannelID  );

//   Use your client's HWND as the wParam.

if( !::PostMessage( HWND_BROADCAST,
                    s_uMCTGRequestMessage,
                    (WPARAM) m_hWnd,
                    lparamSignalIDs        ) )
{
   AfxMessageBox("Failed to request telegraph packet!");
}
```

If there is a MultiClamp telegraph server available to provide information about the specified device / channel, you will receive a WM_COPYDATA message containing an MC_TELEGRAPH_DATA structure for the MultiClamp device / channel you have specified. (see "Message Screening for WM_COPYDATA" below) If you do not receive this message shortly (within say, 1 sec) you should assume that there is no telegraph server for your request.

You will receive no further messages until you issue another request.

Handling a Reconnect Request

The MultiClamp Commander (telegraph server) will broadcast the s_uMCTGReconnectMessage on initialization. The reason for this is to provide a mechanism for existing telegraph clients to reestablish their connections without any user intervention in the event that the server has shut down and restarted while they were connected. This message has as its lParam the signal IDs of the requesting server packed in the format described above. Below is some example code for a client's message loop illustrating how to handle this message. This message should be handled only by those clients which use the s_uMCTGOpenMessage to connect to a telegraph server. Clients which exclusively use s_uMCTGRequestMessage to poll the server need not respond.

```
// process telegraph reconnect message
if( message == s_uMCTGReconnectMessage )
{
   if(!m_bIsConnected)
   {
      // there is no telegraph connection to reestablish
      // ignore this request
      return TRUE;
   }

   // MultiClamp Commander 1.0 (700A)
   if(!MTCG_Match700ASignalIDs( m_mctdCurrentState.uComPortID,
                                m_mctdCurrentState.uAxoBusID,
                                m_mctdCurrentState.uChannelID,
                                lParam                          ) )
   {
      // this request is from a MultiClamp device / channel
      // other than the one we are connected to
      // ignore it
      return TRUE;
   }

   // MultiClamp Commander 2.0 (700B)
   UINT uSerialNum = atoi(m_mctdCurrentState.szSerialNumber);
   UINT uChannelID = m_mctdCurrentState.uChannelID;
   if(!MTCG_Match700BSignalIDs( uSerialNum,
                                uChannelID,
                                lParam     ) )
   {
      // this request is from a MultiClamp device / channel
      // other than the one we are connected to
      // ignore it
      return TRUE;
   }

   // resend the open message to reestablish the connection
   // to the requesting server
   m_bIsConnected = FALSE;
   OnConnect();
   return TRUE;
}
```

Message Screening for WM_COPYDATA

It is very important to verify that a WM_COPYDATA message is the one you are looking for before handling it.

Is this WM_COPYDATA message a MultiClamp telegraph message ?

If so, the dwData member will be equal to the registered message ID you obtained from MCTG_REQUEST_MESSAGE_STR. Note that the size of the received packet depends on which struct version MultiClamp Commander is sending. MultiClamp Commander 2.0 for MultiClamp 700B sends a larger packet than MultiClamp Commander 1.1 for MultiClamp 700A. The following example will ensure that data is always handled correctly regardless of the struct version.

```
COPYDATASTRUCT* pcpds = (COPYDATASTRUCT*) lParam;
if( (pcpds->dwData == (DWORD) s_uMCTGRequestMessage) )
{
   // cast pointer to a telegraph packet
   MC_TELEGRAPH_DATA* pmctdReceived = (MC_TELEGRAPH_DATA*) pcpds->lpData;

   // this WM_COPYDATA message contains a MC_TELEGRAPH_DATA
   // if copying a 700A struct into a 700B struct we copy everything being sent and
   // then leave the rest of the structure as zero. if we are copying a 700B struct
   // into a 700A struct we copy up to the size of a 700A struct and truncate the rest
   // of the sent structure.
   MC_TELEGRAPH_DATA mctdCopied; // constructor will zero structure
   memcpy( &mctdCopied,
           pmctdReceived,
           min(pmctdReceived->uStructSize, sizeof(mctdCopied)) );
}
```

Is the message from the MultiClamp device / channel we want ?
Simply check the identifiers in the MC_TELEGRAPH_DATA structure.

```
MC_TELEGRAPH_DATA* pmctdReceived    = (MC_TELEGRAPH_DATA*) pcpds->lpData;

// MultiClamp Commander 1.1 (700A)

// is it on our device / channel ?
if( ( pmctdReceived->uComPortID != m_mctdCurrentState.uComPortID ) ||
    ( pmctdReceived->uAxoBusID  != m_mctdCurrentState.uAxoBusID  ) ||
    ( pmctdReceived->uChannelID != m_mctdCurrentState.uChannelID )    )
{
   // this message is from another MultiClamp device / channel
   // ignore it
   return TRUE;
}
else
{
  // this is what we are looking for
  // do stuff here
}

// MultiClamp Commander 2.0 (700B)

// is it the correct serial number
if( strncmp( pmctdReceived->szSerialNumber,
             m_mctdCurrentState.szSerialNumber,
             sizeof(m_mctdCurrentState.szSerialNumber) ) != 0 )
{
   // this message is from another MultiClamp device so ignore it
   return TRUE;
}

// is it on our channel ?
if( pmctdReceived->uChannelID != m_mctdCurrentState.uChannelID )
{
   // this message is from another MultiClamp channel so ignore it
   return TRUE;
}

// if we get here we have the correct device
// do stuff here
```

MFC Reference Project - MCTeleClient

This MultiClamp Commander Telegraph SDK contains an MFC reference project called MCTeleClient. This project implements all of the operations outlined in this document, and illustrates how to make use of the other features of the API that have not been addressed here. In particular, the reference code will shed light on the following additional topics:

- **Interpretation and Display of MultiClamp Commander Telegraph Parameters**
- **Populating a combo box with serial numbers from each available server**
- **Making a single connection from many available servers**
- **WM_TIMER – Based Handshaking Scheme For Open and Telegraph Request Operations**

Further details are documented in the MultiClamp Commander Telegraph Interface header file, MCTelegraphs.hpp.