MultiClamp 700x Commander Application Note

How to Control the MultiClamp 700x from Another Application

Last revision: June 4, 2004

MultiClamp 700x Commander can be controlled from another application with a special API. This Application Note provides some basic guidance in how to do this.

Note that we cannot offer technical support on this matter, and the API may change without warning.

Use this document in conjunction with the header file AxMultiClampMsg.h and example VC++ code in the MCCMSG_TestBed project.

Before calling any command, you must first create the DLL object. `MCCMSG_CreateObject` returns a handle that you must pass as a parameter to each subsequent call. The handle references a hidden window inside the DLL for capturing Windows messages. Make sure you also destroy the DLL object using `MCCMSG_DestroyObject` before exiting your application. This will prevent a memory leak.

After creating the DLL object, you must select the MultiClamp 700x Commander and headstage channel you wish to control. This is done by scanning for all MultiClamp 700x devices connected to a MultiClamp 700x Commander.

In this discussion, a device is defined as one Multiclamp 700x amplifier with one headstage channel controlled by MultiClamp 700x Commander. A MultiClamp 700x amplifier therefore supports two devices when both headstages are plugged in.

`MCCMSG_FindFirstMultiClamp` and `MCCMSG_FindNextMultiClamp` uniquely identify each device based on the amplifier model, the headstage channel, COM port information (model 700A) or the serial number (model 700B). If your rig has just one headstage, you won't need to call `MCCMSG_FindNextMultiClamp` because the first found device is always the correct device. Once you have found the correct device, use `MCCMSG_SelectMultiClamp` to select the device you wish to control. The C++ example at the end of this section shows how to do this.

Error handling is standard. A command always returns FALSE if the error parameter is set. The error can be decoded by passing the returned error code `*pnError` to `MCCMSG_BuildErrorText`.

Inside the DLL, each command is based on the Windows messaging technique. This means that commands can potentially timeout if the system is busy. The default timeout (3 second) is sufficient for all commands. To change the timeout value call `MCCMSG_SetTimeOut`. In the event of a timeout, a command will block for the timeout period and return FALSE with timeout error code `MCCMSG_ERROR_MSGTIMEOUT`.

Some functions such as `MCCMSG_SetMode` and `MCCMSG_SetPrimarySignal` use predefined constants as parameters. e.g. The following code changes the amplifier mode to voltage clamp:

```
int nError = MCCMSG_ERROR_NOERROR;
MCCMSG_SetMode(m_hMCCmsg, MCCMSG_MODE_VCLAMP, &nError);
```

A complete list of predefined constants can be found in AxMultiClampMsg.h.

To get started, compile the VC++ example project MCCMSG_TestBed in DevStudio. Run the test bed and then run MultiClamp 700x Commander. In the test bed, click "Create DLL", then "Scan", then choose an amplifier from the Device combo box. Notice that the MultiClamp 700x Commander changes the active mode/channel tab when the headstage channel is changed. Now select the voltage clamp mode by clicking the "VC" radio button and observe that the MultiClamp 700x Commander will change to that mode.

Now try clicking the Auto Fast Compensation button. When the command completes, check that the Cp Fast and Cp Tau edit boxes are filled out with the same values as MultiClamp 700x Commander. All values are in SI units, so even though these numbers are formatted in pF and µs, the underlying code uses the values in Farads and seconds.

It is important that you setup the correct mode before you apply a command because the DLL sends messages only to the active mode/channel tab in MultiClamp 700x Commander. For example, `MCCMSG_AutoFastComp` will work only when the MultiClamp 700x amplifier is in voltage clamp mode and the active mode/channel tab is voltage clamp. If you now click the current clamp tab, the same command will fail with error `MCCMSG_ERROR_INVALIDPARAMETER`.

Example

```
#include <afxwin.h> // MFC core and standard components
#include "AxMultiClampMsg.h"

//==============================================================================
// FUNCTION: DisplayErrorMsg
// PURPOSE:  Display error as text string
//
void DisplayErrorMsg(HMCCMSG hMCCmsg, int nError)
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}

//==============================================================================
// FUNCTION: main
// PURPOSE:  This example shows how to create the DLL handle, select the first
//           MultiClamp, set current clamp mode, execute auto fast and
//           slow compensation, and destroy the handle.
//
int main()
{
   // check the API version matches the expected value
   if( !MCCMSG_CheckAPIVersion(MCCMSG_APIVERSION_STR) )
   {
      AfxMessageBox("Version mismatch: AXCLAMPEXMSG.DLL", MB_ICONSTOP);
      return 0;
   }

   // create DLL handle
   int nError = MCCMSG_ERROR_NOERROR;
   HMCCMSG hMCCmsg = MCCMSG_CreateObject(&nError);
   if( !hMCCmsg )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // find the first MultiClamp
   char szError[256]    = "";
   char szSerialNum[16] = ""; // Serial number of MultiClamp 700B
   UINT uModel          = 0;  // Identifies MultiClamp 700A or 700B model
   UINT uCOMPortID      = 0;  // COM port ID of MultiClamp 700A (1-16)
   UINT uDeviceID       = 0;  // Device ID of MultiClamp 700A (1-8)
   UINT uChannelID      = 0;  // Headstage channel ID
   if( !MCCMSG_FindFirstMultiClamp(hMCCmsg, &uModel, szSerialNum,
                                   sizeof(szSerialNum), &uCOMPortID,
                                  &uDeviceID, &uChannelID, &nError)  )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // select this MultiClamp
   if( !MCCMSG_SelectMultiClamp(hMCCmsg, uModel, szSerialNum,
                                uCOMPortID, uDeviceID, uChannelID, &nError) )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // set voltage clamp mode
   if( !MCCMSG_SetMode(hMCCmsg, MCCMSG_MODE_VCLAMP, &nError) )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // execute auto fast compensation
   if( !MCCMSG_AutoFastComp(hMCCmsg, &nError) )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // execute auto slow compensation
   if( !MCCMSG_AutoSlowComp(hMCCmsg, &nError) )
   {
      DisplayErrorMsg(hMCCmsg, nError);
      return 0;
   }

   // destroy DLL handle
   MCCMSG_DestroyObject(hMCCmsg);
   hMCCmsg = NULL;

   return 0;
}
```

Command Reference

Commands with similar functions are grouped together. Command groups are ordered as in the MultiClamp 700x Commander GUI from top to bottom and from left to right.

DLL creation and destruction

BOOL MCCMSG_CheckAPIVersion(LPCSTR pszQueryVersion);

Parameter	**Description**
pszQueryVersion	Pointer to null terminated string containing the version number.

Remarks
Checks the API version matches the expected value: `MCCMSG_APIVERSION_STR`.

Example
```
#include "AxMultiClampMsg.h"

if( !MCCMSG_CheckAPIVersion(MCCMSG_APIVERSION_STR) )
{
   AfxMessageBox("Version mismatch: AXMULTICLAMPMSG.DLL", MB_ICONSTOP);
   return;
}
```

HMCCMSG MCCMSG_CreateObject(int *pnError);

Parameter	**Description**
pnError	Address of error return code.

Remarks
Create the MultiClamp 700x Commander message handler object and return a 32-bit handle. When successful, the returned handle is non-NULL. You must use this handle to call other MCCMSG functions.

Error Codes
```
MCCMSG_ERROR_OUTOFMEMORY
MCCMSG_ERROR_MCCNOTOPEN
```

Example
```
#include "AxMultiClampMsg.h"

int nError = MCCMSG_ERROR_NOERROR;
m_hMCCmsg = MCCMSG_CreateObject(&nError);
if( !m_hMCCmsg )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(NULL, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

void MCCMSG_DestroyObject(HMCCMSG hMCCmsg);

Parameter	**Description**
hMCCmsg	Handle to message handler object.

Remarks
Destroys the MultiClamp 700x Commander message handler object. To prevent memory leaks call this before exiting your application.

Example

```
#include "AxMultiClampMsg.h"

MCCMSG_DestroyObject(m_hMCCmsg);
m_hMCCmsg = NULL;
```

General

BOOL MCCMSG_SetTimeOut(HMCCMSG hMCCmsg, UINT uTimeOutMS, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uTimeOutMS	Time out value in milliseconds.
pnError	Address of error return code.

Remarks

Set the time out value for Windows messages between the message handler object and MultiClamp 700x Commander. Default time out is 3 second.

Example

```
#include "AxMultiClampMsg.h"

UINT uTimeOut = 2000; // 2 seconds
if( !MCCMSG_SetTimeOut(m_hMCCmsg, uTimeOut, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

MultiClamp 700x Commander selection

BOOL MCCMSG_FindFirstMultiClamp(HMCCMSG hMCCmsg, UINT *puModel, char *pszSerialNum, UINT uBufSize, UINT *puCOMPortID, UINT *puDeviceID, UINT *puChannelID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
puModel	Address of return amplifier model ID.
pszSerialNum	Address of return serial number (MultiClamp 700B only).
uBufSize	Size of char buffer for serial number (MultiClamp 700B only).
puCOMPortID	Address of return COM port ID (MultiClamp 700A only)
puDeviceID	Address of return Device ID (MultiClamp 700A only)
puChannelID	Address of return Channel ID
pnError	Address of error return code.

Remarks
Find the first open MultiClamp 700x amplifier. At least one MultiClamp 700x Commander must be open to return valid parameters.

Inside the DLL, `MCCMSG_FindFirstMultiClamp` scans and stores all open devices but only returns parameters for the first. `MCCMSG_FindNextMultiClamp` uses this internally stored data to access information on each device. You must therefore call `MCCMSG_FindFirstMultiClamp` before calling `MCCMSG_FindNextMultiClamp`. Note that you don't have to call `MCCMSG_FindNextMultiClamp` if your rig has just one headstage. In this case, `MCCMSG_FindFirstMultiClamp` will always find the correct device.

If you close and reopen a MultiClamp 700x Commander after executing `MCCMSG_FindFirstMultiClamp` you must refresh the internal list held by the DLL by executing `MCCMSG_FindFirstMultiClamp` again.

This command supports the MultiClamp 700A and the MultiClamp 700B. When a MultiClamp 700A is detected, `*puModel` is set to `MCCMSG_HW_TYPE_MC700A` and `*puCOMPortID` and `*puDeviceID` are filled out (`*pszSerialNum` is not used). When a MultiClamp 700B is detected, `*puModel` is set to `MCCMSG_HW_TYPE_MC700B` and `*pszSerialNum` is filled out (`*puCOMPortID` and `*puDeviceID` are not used).

Error Codes

`MCCMSG_ERROR_MCCNOTOPEN`	MultiClamp 700x Commander not open

Example
```
#include "AxMultiClampMsg.h"

char szError[256]     = "";
UINT uModel           = 0;  // Identifies MultiClamp 700A or 700B model
char szSerialNum[16] = ""; // Serial number of MultiClamp 700B
UINT uCOMPortID       = 0;  // COM port ID of MultiClamp 700A (1-16)
UINT uDeviceID        = 0;  // Device ID of MultiClamp 700A (1-8)
UINT uChannelID       = 0;  // Headstage channel ID

// find the first MultiClamp
if( !MCCMSG_FindFirstMultiClamp(m_hMCCmsg, &uModel, szSerialNum,
                                 sizeof(szSerialNum), &uCOMPortID,
                                &uDeviceID, &uChannelID, &nError)  )
{
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
   return;
}

// create string with device info
char szDeviceInfo[256] = "";
switch( uModel )
{
   case MCCMSG_HW_TYPE_MC700A:
      sprintf(szDeviceInfo, "MultiClamp 700A open on COM port %d,
              device ID %d. Headstage channel %d connected.",
              uCOMPortID, uDeviceID, uChannelID);
      break;
   case MCCMSG_HW_TYPE_MC700B:
      sprintf(szDeviceInfo, "MultiClamp 700B open on USB port
              with serial number %s. Headstage channel %d connected.",
              szSerialNum, uChannelID);
}

// display device info
AfxMessageBox(szDeviceInfo, MB_ICONINFORMATION);
```

BOOL MCCMSG_FindNextMultiClamp(HMCCMSG hMCCmsg, UINT *puModel, char *pszSerialNum, UINT uBufSize, UINT *puCOMPortID, UINT *puDeviceID, UINT *puChannelID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
puModel	Address of return amplifier model ID.
pszSerialNum	Address of return serial number (MultiClamp 700B only).
uBufSize	Size of char buffer for serial number (MultiClamp 700B only).
puCOMPortID	Address of return COM port ID (MultiClamp 700A only).
puDeviceID	Address of return Device ID (MultiClamp 700A only).
puChannelID	Address of return Channel ID.
pnError	Address of error return code.

Remarks
Find the next open MultiClamp 700x amplifier. `MCCMSG_FindFirstMultiClamp` must be called first to create an internal list of devices. Returns FALSE when all devices have been found.

This command supports the MultiClamp 700A and the MultiClamp 700B. When a MultiClamp 700A is detected, `*puModel` is set to `MCCMSG_HW_TYPE_MC700A` and `*puCOMPortID` and `*puDeviceID` are filled out (`*pszSerialNum` is not used). When a MultiClamp 700B is detected, `*puModel` is set to `MCCMSG_HW_TYPE_MC700B` and `*pszSerialNum` is filled out (`*puCOMPortID` and `*puDeviceID` are not used).

Example
```
#include "AxMultiClampMsg.h"

// find the first MultiClamp
char szSerialNum[16] = ""; // Serial number of MultiClamp 700B
UINT uModel          = 0;  // Identifies MultiClamp 700A or 700B model
UINT uCOMPortID      = 0;  // COM port ID of MultiClamp 700A (1-16)
UINT uDeviceID       = 0;  // Device ID of MultiClamp 700A (1-8)
UINT uChannelID      = 0;  // Headstage channel ID
if( !MCCMSG_FindFirstMultiClamp(hMCCmsg, &uModel, szSerialNum,
                                sizeof(szSerialNum), &uCOMPortID,
                               &uDeviceID, &uChannelID, &nError) )
{
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
   return;
}

// find the second MultiClamp, third MultiClamp, ... etc
while(1)
{
   // create string with device info
   char szDeviceInfo[256] = "";
   switch( uModel )
   {
      case MCCMSG_HW_TYPE_MC700A:
         sprintf(szDeviceInfo, "MultiClamp 700A open on COM port %d,
                 device ID %d. Headstage channel %d connected.",
                 uCOMPortID, uDeviceID, uChannelID);
         break;
      case MCCMSG_HW_TYPE_MC700B:
         sprintf(szDeviceInfo, "MultiClamp 700B open on USB port
                 with serial number %s. Headstage channel %d connected.",
                 szSerialNum, uChannelID);
   }

   // display device info
   AfxMessageBox(szDeviceInfo, MB_ICONINFORMATION);

   // find the next MultiClamp
   if( !MCCMSG_FindNextMultiClamp(hMCCmsg, &uModel, szSerialNum,
                                  sizeof(szSerialNum), &uCOMPortID,
                                  &uDeviceID, &uChannelID, &nError) )
   {
      break;
   }
}
```

BOOL MCCMSG_SelectMultiClamp(HMCCMSG hMCCmsg, UINT uModel, char *pszSerialNum, UINT uCOMPortID, UINT uDeviceID, UINT uChannelID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uModel	The amplifier model ID.
pszSerialNum	The serial number (MultiClamp 700B only).
uCOMPortID	The COM port ID (MultiClamp 700A only).
uDeviceID	The Device ID (MultiClamp 700A only).
uChannelID	The Channel ID.
pnError	Address of error return code.

Remarks
Select the MultiClamp 700x amplifier to control. You must call `MCCMSG_FindFirstMultiClamp` before using this command. Use `MCCMSG_FindFirstMultiClamp` and/or `MCCMSG_FindNextMultiClamp` to check that a device is available before calling `MCCMSG_SelectMultiClamp`. Returns FALSE if the specified device is not open.

This command supports the MultiClamp 700A and the MultiClamp 700B. To connect to a MultiClamp 700A, set `uModel` to `MCCMSG_HW_TYPE_MC700A` and fill in `uCOMPortID` and `uDeviceID` (`szSerialNum` is not used). To connect to a MultiClamp 700B, set `uModel` to `MCCMSG_HW_TYPE_MC700B` and fill in `szSerialNum` (`uCOMPortID` and `uDeviceID` are not used).

Error Codes

`MCCMSG_ERROR_MCCNOTOPEN`	MultiClamp 700x Commander not open

Example
```
#include "AxMultiClampMsg.h"

char szError[256]    = "";
UINT uModel          = 0;  // Identifies MultiClamp 700A or 700B model
char szSerialNum[16] = ""; // Serial number of MultiClamp 700B
UINT uCOMPortID      = 0;  // COM port ID of MultiClamp 700A (1-16)
UINT uDeviceID       = 0;  // Device ID of MultiClamp 700A (1-8)
UINT uChannelID      = 0;  // Headstage channel ID

// find the first MultiClamp
if( !MCCMSG_FindFirstMultiClamp(m_hMCCmsg, &uModel, szSerialNum,
                                sizeof(szSerialNum), &uCOMPortID,
                               &uDeviceID, &uChannelID, &nError) )
{
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
   return;
}

// select the first MultiClamp
if( !MCCMSG_SelectMultiClamp(m_hMCCmsg, uModel, szSerialNum,
                             uCOMPortID, uDeviceID, uChannelID, &nError) )
{
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
   return;
}
```

Amplifier Mode

BOOL MCCMSG_SetMode(HMCCMSG hMCCmsg, UINT uModeID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uModeID	The amplifier mode.
pnError	Address of error return code.

Remarks
Set the amplifier mode to voltage clamp, current clamp or I=0. The MultiClamp 700x Commander mode radio button, and the active tab will switch to the selected mode if successful. This command must precede other commands to ensure the amplifier is in the correct mode. `uModeID` must be filled out with:

`MCCMSG_MODE_VCLAMP`	Voltage clamp mode
`MCCMSG_MODE_ICLAMP`	Current clamp mode
`MCCMSG_MODE_ICLAMPZERO`	I=0 mode

Example
```
#include "AxMultiClampMsg.h"

// set mode to voltage clamp
if( !MCCMSG_SetMode(m_hMCCmsg, MCCMSG_MODE_VCLAMP, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetMode(HMCCMSG hMCCmsg, UINT *puModeID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
puModeID	Address of return amplifier mode.
pnError	Address of error return code.

Remarks
Get the current amplifier mode. `*puModeID` will be filled in with:

`MCCMSG_MODE_VCLAMP`	Voltage clamp mode
`MCCMSG_MODE_ICLAMP`	Current clamp mode
`MCCMSG_MODE_ICLAMPZERO`	I=0 mode

Example
```
#include "AxMultiClampMsg.h"

// get the current mode
UINT uMode = 0;
if( !MCCMSG_GetMode(m_hMCCmsg, &uMode, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetModeSwitchEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The mode switch enable state.
pnError	Address of error return code.

Remarks
Set the amplifier mode switch enable state. This controls the checkbox on the right hand side of the mode radio button. When set to TRUE, the MultiClamp 700x Commander mode buttons can be operated from an external TTL source (ext) or can be triggered on a preset membrane potential (700B only). These options must be manually setup from the Options / Auto tab (700B only).

Example
```
#include "AxMultiClampMsg.h"

// set the mode switch enable state
BOOL bEnable = TRUE;
if( !MCCMSG_SetModeSwitchEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetModeSwitchEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return mode switch enable state.
pnError	Address of error return code.

Remarks
This gets the state of the checkbox on the right hand side of the mode radio button. When TRUE, the MultiClamp 700x Commander mode buttons can be operated from an external TTL source (ext) or can be triggered on a preset membrane potential (700B only). These options must be manually setup from the Options / Auto tab (700B only).

Example
```
#include "AxMultiClampMsg.h"

// get the mode switch enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetModeSwitchEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Holding

BOOL MCCMSG_SetHoldingEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The holding enable state.
pnError	Address of error return code.

Remarks
Set the holding enable state. The holding level is applied to the hardware when TRUE. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the holding enable state
BOOL bEnable = TRUE;
if( !MCCMSG_SetHoldingEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetHoldingEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return holding enable state.
pnError	Address of error return code.

Remarks
Get the holding enable state. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the holding enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetHoldingEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetHolding(HMCCMSG hMCCmsg, BOOL dHolding, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dHolding	The holding level in SI units (Volts or Amps).
pnError	Address of error return code.

Remarks
Set the holding level. Supports voltage clamp and current clamp. The holding level must be in SI units appropriate to the amplifier mode.

Example
```
#include "AxMultiClampMsg.h"

// set the holding level
double dHolding = 0;
if( !MCCMSG_SetHolding(m_hMCCmsg, dHolding, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetHolding(HMCCMSG hMCCmsg, BOOL *pdHolding, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdHolding	Address of return holding level in SI units (Volts or Amps).
pnError	Address of error return code.

Remarks
Get the holding level. Supports voltage clamp and current clamp. The return holding level will be in SI units appropriate to the amplifier mode.

Example
```
#include "AxMultiClampMsg.h"

// get the holding level
double dHolding = 0;
if( !MCCMSG_GetHolding(m_hMCCmsg, &dHolding, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Test Signal

BOOL MCCMSG_SetTestSignalEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The test signal enable state.
pnError	Address of error return code.

Remarks
Set the test signal enable. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the test signal enable state
BOOL dEnable = TRUE;
if( !MCCMSG_SetTestSignalEnable(m_hMCCmsg, dEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetTestSignalEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return test signal enable state.
pnError	Address of error return code.

Remarks
Get the test signal enable. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the test signal enable state
BOOL dEnable = FALSE;
if( !MCCMSG_GetTestSignalEnable(m_hMCCmsg, &dEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetTestSignalAmplitude(HMCCMSG hMCCmsg, double dAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dAmplitude	The test signal amplitude (Volts or Amps).
pnError	Address of error return code.

Remarks
Set the test signal amplitude in Volts or Amps. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp. The amplitude must be in SI units appropriate to the amplifier mode.

Example
```
#include "AxMultiClampMsg.h"

// set the test signal amplitude
double dAmplitude = 0;
if( !MCCMSG_SetTestSignalAmplitude(m_hMCCmsg, dEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetTestSignalAmplitude(HMCCMSG hMCCmsg, double *pdAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdAmplitude	Address of return test signal amplitude (Volts or Amps).
pnError	Address of error return code.

Remarks
Get the test signal amplitude in Volts or Amps. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp. The return value will be in SI units appropriate to the amplifier mode.

Example
```
#include "AxMultiClampMsg.h"

// get the test signal amplitude
double dAmplitude = 0;
if( !MCCMSG_GetTestSignalAmplitude(m_hMCCmsg, &dAmplitude, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetTestSignalFrequency(HMCCMSG hMCCmsg, double dFrequency, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dFrequency	The test signal frequency (Hertz).
pnError	Address of error return code.

Remarks
Set the test signal amplitude in Hertz. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the test signal frequency
double dFrequency = 0;
if( !MCCMSG_SetTestSignalFrequency(m_hMCCmsg, dFrequency, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetTestSignalFrequency(HMCCMSG hMCCmsg, double *pdFrequency, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdFrequency	Address of return test signal frequency (Hertz).
pnError	Address of error return code.

Remarks
Get the test signal amplitude in Hertz. Supports voltage clamp and current clamp. The test signal control is called "Seal Test" in voltage clamp and "Tuning" in current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the test signal frequency
double dFrequency = 0;
if( !MCCMSG_GetTestSignalFrequency(m_hMCCmsg, &dFrequency, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Pipette Offset

BOOL MCCMSG_AutoPipetteOffset(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic pipette offset. This calculates and compensates for potentials caused by differences in the concentration of bath and pipette solutions. A timeout value of 3 seconds is recommended for this command. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto pipette offset
if( !MCCMSG_AutoPipetteOffset(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPipetteOffset(HMCCMSG hMCCmsg, double dPipetteOffset, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dPipetteOffset	The pipette offset (Volts).
pnError	Address of error return code.

Remarks
Set the pipette offset in Volts. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the pipette offset
double dPipetteOffset = 0.01; // 10 mV
if( !MCCMSG_SetPipetteOffset(m_hMCCmsg, dPipetteOffset, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPipetteOffset(HMCCMSG hMCCmsg, double *pdPipetteOffset, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdPipetteOffset	Address of return pipette offset (Volts).
pnError	Address of error return code.

Remarks
Get the pipette offset in Volts. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the pipette offset
double dPipetteOffset = 0;
if( !MCCMSG_GetPipetteOffset(m_hMCCmsg, &dPipetteOffset, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Slow Current Injection

BOOL MCCMSG_SetSlowCurrentInjEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The slow current inject enable state.
pnError	Address of error return code.

Remarks
Set the slow current inject enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable slow current inject
BOOL bEnable = TRUE;
if( !MCCMSG_SetSlowCurrentInjEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCurrentInjEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return slow current inject enable state.
pnError	Address of error return code.

Remarks
Get the slow current inject enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the slow current inject enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetSlowCurrentInjEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSlowCurrentInjLevel(HMCCMSG hMCCmsg, double dLevel, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dLevel	The slow current injection level (Volts).
pnError	Address of error return code.

Remarks
Set the slow current injection level in Volts. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the slow current injection level
BOOL dLevel = 0.06; // 60 mV
if( !MCCMSG_SetSlowCurrentInjLevel(m_hMCCmsg, dLevel, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCurrentInjLevel(HMCCMSG hMCCmsg, double *pdLevel, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdLevel	Address of return slow current injection level (Volts).
pnError	Address of error return code.

Remarks
Get the slow current injection level in Volts. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the slow current injection level
BOOL dLevel = 0;
if( !MCCMSG_GetSlowCurrentInjLevel(m_hMCCmsg, &dLevel, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSlowCurrentInjSettlingTime(HMCCMSG hMCCmsg, double dSettlingTime, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dSettlingTime	The slow current injection settling time (seconds).

pnError	Address of error return code.

Remarks
Set the slow current injection settling time in seconds. Settling time is defined as the time to 99% of the final value when the feedback resistor equals the resistive load. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the slow current injection settling time
BOOL dSettlingTime = 1; // 1 second
if( !MCCMSG_SetSlowCurrentInjSettlingTime(m_hMCCmsg, dSettlingTime, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCurrentInjSettlingTime(HMCCMSG hMCCmsg, double *pdSettlingTime, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdSettlingTime	Address of return slow current injection settling time (seconds).
pnError	Address of error return code.

Remarks
Get the slow current injection settling time in seconds. Settling time is defined as the time to 99% of the final value when the feedback resistor equals the resistive load. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the slow current injection settling time
BOOL dSettlingTime = 0;
if( !MCCMSG_GetSlowCurrentInjSettlingTime(m_hMCCmsg, &dSettlingTime, &nError)
)
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Fast and Slow Compensation

BOOL MCCMSG_SetFastCompCap(HMCCMSG hMCCmsg, double dFastCompCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dFastCompCap	The fast compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Set the fast compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the fast compensation capacitance
double dFastCompCap = 4e-12; // 4 pF
if( !MCCMSG_SetFastCompCap(m_hMCCmsg, dFastCompCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetFastCompCap(HMCCMSG hMCCmsg, double *pdFastCompCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdFastCompCap	Address of return fast compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Get the fast compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the fast compensation capacitance
double dFastCompCap = 0;
if( !MCCMSG_GetFastCompCap(m_hMCCmsg, &dFastCompCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSlowCompCap(HMCCMSG hMCCmsg, double dSlowCompCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dSlowCompCap	The slow compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Set the slow compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the slow compensation capacitance
double dSlowCompCap = 3e-12; // 3 pF
if( !MCCMSG_SetSlowCompCap(m_hMCCmsg, dSlowCompCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCompCap(HMCCMSG hMCCmsg, double *pdSlowCompCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdSlowCompCap	Address of return slow compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Get the slow compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the slow compensation capacitance
double dSlowCompCap = 0;
if( !MCCMSG_GetSlowCompCap(m_hMCCmsg, &dSlowCompCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetFastCompTau(HMCCMSG hMCCmsg, double dFastCompTau, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dFastCompTau	The fast compensation time constant (seconds).

pnError	Address of error return code.

Remarks
Set the fast compensation time constant in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the fast compensation time constant
double dFastCompTau = 2e-6; // 2 µs
if( !MCCMSG_SetFastCompTau(m_hMCCmsg, dFastCompTau, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetFastCompTau(HMCCMSG hMCCmsg, double *pdFastCompTau, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdFastCompTau	Address of return fast compensation time constant (seconds).
pnError	Address of error return code.

Remarks
Get the fast compensation time constant in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the fast compensation time constant
double dFastCompTau = 0;
if( !MCCMSG_GetFastCompTau(m_hMCCmsg, &dFastCompTau, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSlowCompTau(HMCCMSG hMCCmsg, double dSlowCompTau, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dSlowCompTau	The slow compensation time constant (seconds).
pnError	Address of error return code.

Remarks
Set the slow compensation time constant in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the slow compensation time constant
double dSlowCompTau = 4e-4; // 400 μs
if( !MCCMSG_SetSlowCompTau(m_hMCCmsg, dSlowCompTau, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCompTau(HMCCMSG hMCCmsg, double *pdSlowCompTau, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdSlowCompTau	Address of return slow compensation time constant (seconds).
pnError	Address of error return code.

Remarks
Get the slow compensation time constant in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the slow compensation time constant
double dSlowCompTau = 0;
if( !MCCMSG_GetSlowCompTau(m_hMCCmsg, &dSlowCompTau, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSlowCompTauX20Enable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The tau x20 range enable state.
pnError	Address of error return code.

Remarks
Set the slow compensation tau x20 range enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable the tau x20 range
BOOL bEnable = TRUE;
if( !MCCMSG_SetSlowCompTauX20Enable(m_hMCCmsg, bEnable, &nError) )
```

```
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSlowCompTauX20Enable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return tau x20 range enable state.
pnError	Address of error return code.

Remarks
Get the slow compensation tau x20 range enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the tau x20 range enable state
BOOL bEnable = FASLE;
if( !MCCMSG_SetSlowCompTauX20Enable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoFastComp(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic fast compensation. This calculates and compensates for fast pipette capacitance. A timeout value of 3 seconds is recommended for this command. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto fast compensation
if( !MCCMSG_AutoFastComp(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoSlowComp(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic slow compensation. This calculates and compensates for slow pipette capacitance. A timeout value of 3 seconds is recommended for this command. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto slow compensation
if( !MCCMSG_AutoSlowComp(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Pipette Capacitance Neutralization

BOOL MCCMSG_SetNeutralizationEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The pipette capacitance neutralization enable state.
pnError	Address of error return code.

Remarks
Set the pipette capacitance neutralization enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable pipette capacitance neutralization
BOOL bEnable = TRUE;
if( !MCCMSG_SetNeutralizationEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetNeutralizationEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return pipette capacitance neutralization enable state.
pnError	Address of error return code.

Remarks
Get the pipette capacitance neutralization enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the pipette capacitance neutralization enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetNeutralizationEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetNeutralizationCap(HMCCMSG hMCCmsg, double dCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dCap	The pipette neutralization capacitance (Farads).
pnError	Address of error return code.

Remarks
Set the pipette neutralization capacitance in Farads. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set pipette neutralization capacitance
double dCap = 4e-12; // 4 pF
if( !MCCMSG_SetNeutralizationCap(m_hMCCmsg, dCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetNeutralizationCap(HMCCMSG hMCCmsg, double *pdCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdCap	Address of return pipette neutralization capacitance (Farads).
pnError	Address of error return code.

Remarks
Get the pipette neutralization capacitance in Farads. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get pipette neutralization capacitance
double dCap = 0;
if( !MCCMSG_GetNeutralizationCap(m_hMCCmsg, &dCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Whole Cell Compensation

BOOL MCCMSG_SetWholeCellCompEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The whole cell compensation enable state.
pnError	Address of error return code.

Remarks
Set the whole cell compensation enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable whole cell compensation
BOOL bEnable = TRUE;
if( !MCCMSG_SetWholeCellCompEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetWholeCellCompEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return whole cell compensation enable state.
pnError	Address of error return code.

Remarks
Get the whole cell compensation enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the whole cell compensation enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetWholeCellCompEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetWholeCellCompCap(HMCCMSG hMCCmsg, double dCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dCap	The whole cell compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Set the whole cell compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the whole cell slow compensation capacitance
double dCap = 3e-11; // 30 pF
if( !MCCMSG_SetWholeCellCompCap(m_hMCCmsg, dCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetWholeCellCompCap(HMCCMSG hMCCmsg, double *pdCap, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdCap	Address of return whole cell compensation capacitance (Farads).
pnError	Address of error return code.

Remarks
Get the whole cell compensation capacitance in Farads. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the whole cell slow compensation capacitance
double dCap = 0;
if( !MCCMSG_GetWholeCellCompCap(m_hMCCmsg, &dCap, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetWholeCellCompResist(HMCCMSG hMCCmsg, double dResist, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dResist	The whole cell compensation resistance (Ohms).
pnError	Address of error return code.

Remarks
Set the whole cell compensation resistance in Ohms. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the whole cell compensation resistance
double dResist = 1e13; // 10 MOhms
if( !MCCMSG_SetWholeCellCompResist(m_hMCCmsg, dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetWholeCellCompResist(HMCCMSG hMCCmsg, double *pdResist, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdResist	Address of return whole cell compensation resistance (Ohms).
pnError	Address of error return code.

Remarks
Get the whole cell compensation resistance in Ohms. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the whole cell compensation resistance
double dResist = 0;
if( !MCCMSG_GetWholeCellCompResist(m_hMCCmsg, &dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoWholeCellComp(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic whole cell compensation. This calculates and compensates for cell membrane capacitance in whole cell experiments. A timeout value of 3 seconds is recommended for this command. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto whole cell compensation
```

```
if( !MCCMSG_AutoWholeCellComp(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Series Resistance Compensation

BOOL MCCMSG_SetRsCompEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The series resistance compensation enable state.
pnError	Address of error return code.

Remarks
Set the series resistance compensation enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable series resistance compensation
BOOL bEnable = TRUE;
if( !MCCMSG_SetRsCompEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetRsCompEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return series resistance compensation enable state.
pnError	Address of error return code.

Remarks
Get the series resistance compensation enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the series resistance compensation enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetRsCompEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetRsCompBandwidth(HMCCMSG hMCCmsg, double dBandwidth, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dBandwidth	The series resistance compensation bandwidth in Hertz.
pnError	Address of error return code.

Remarks
Set the series resistance compensation bandwidth in Hertz. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the series resistance compensation bandwidth
double dBandwidth = 1e4; // 10 kHz
if( !MCCMSG_SetRsCompBandwidth(m_hMCCmsg, dBandwidth, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetRsCompBandwidth(HMCCMSG hMCCmsg, double *pdBandwidth, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdBandwidth	Address of return series resistance compensation bandwidth in Hertz.
pnError	Address of error return code.

Remarks
Set the series resistance compensation bandwidth in Hertz. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the series resistance compensation bandwidth
double dBandwidth = 0;
if( !MCCMSG_GetRsCompBandwidth(m_hMCCmsg, &dBandwidth, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetRsCompCorrection(HMCCMSG hMCCmsg, double dCorrection, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.

dCorrection	The series resistance compensation correction (%).
pnError	Address of error return code.

Remarks
Set the series resistance compensation correction (%). Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the series resistance compensation correction
double dCorrection = 80; // 80%
if( !MCCMSG_SetRsCompCorrection(m_hMCCmsg, dCorrection, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetRsCompCorrection(HMCCMSG hMCCmsg, double *pdCorrection, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dCorrection	Address of return series resistance compensation correction (%).
pnError	Address of error return code.

Remarks
Get the series resistance compensation correction (%). Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the series resistance compensation correction
double dCorrection = 0;
if( !MCCMSG_GetRsCompCorrection(m_hMCCmsg, &dCorrection, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetRsCompPrediction(HMCCMSG hMCCmsg, double dPrediction, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dPrediction	The series resistance compensation prediction (%).
pnError	Address of error return code.

Remarks
Set the series resistance compensation prediction (%). Supports voltage clamp.

Example

```
#include "AxMultiClampMsg.h"

// set the series resistance compensation prediction
double dPrediction = 80; // 80%
if( !MCCMSG_SetRsCompPrediction(m_hMCCmsg, dPrediction, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetRsCompPrediction(HMCCMSG hMCCmsg, double *pdPrediction, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dPrediction	Address of return series resistance compensation prediction (%).
pnError	Address of error return code.

Remarks

Get the series resistance compensation prediction (%). Supports voltage clamp.

Example

```
#include "AxMultiClampMsg.h"

// get the series resistance compensation prediction
double dPrediction = 0;
if( !MCCMSG_GetRsCompPrediction(m_hMCCmsg, &dPrediction, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Oscillation Killer

BOOL MCCMSG_SetOscKillerEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The oscillator killer enable state.
pnError	Address of error return code.

Remarks
Set the oscillator killer enable state. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable the oscillator killer
BOOL bEnable = TRUE;
if( !MCCMSG_SetOscKillerEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetOscKillerEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return oscillator killer enable state.
pnError	Address of error return code.

Remarks
Get the oscillator killer enable state. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the oscillator killer enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetOscKillerEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Primary (or Scaled) Signal

BOOL MCCMSG_SetPrimarySignal(HMCCMSG hMCCmsg, UINT uSignalID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uSignalID	The primary output signal.
pnError	Address of error return code.

Remarks
Set the primary output signal. Supports voltage and current clamp. `uSignalID` must be filled in as one of the following values:

`MCCMSG_PRI_SIGNAL_VC_MEMBCURRENT`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_MEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_PIPPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_100XACMEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_EXTCMDPOTENTIAL`	voltage clamp, 700B only
`MCCMSG_PRI_SIGNAL_VC_AUXILIARY1`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_AUXILIARY2`	voltage clamp, 700B only
`MCCMSG_PRI_SIGNAL_IC_MEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_MEMBCURRENT`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_CMDCURRENT`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_100XACMEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_EXTCMDCURRENT`	current clamp, 700B only
`MCCMSG_PRI_SIGNAL_IC_AUXILIARY1`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_AUXILIARY2`	current clamp, 700B only

Example
```
#include "AxMultiClampMsg.h"

// set the primary output to external command potential in voltage clamp
UINT uSignalID = MCCMSG_PRI_SIGNAL_VC_EXTCMDPOTENTIAL;
if( !MCCMSG_SetPrimarySignal(m_hMCCmsg, uSignalID, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPrimarySignal(HMCCMSG hMCCmsg, UINT *puSignalID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
puSignalID	Address of return primary output signal.
pnError	Address of error return code.

Remarks
Get the primary output signal. Supports voltage and current clamp. `uSignalID` will be filled out as one of the following values:

`MCCMSG_PRI_SIGNAL_VC_MEMBCURRENT`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_MEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_PIPPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_100XACMEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_EXTCMDPOTENTIAL`	voltage clamp, 700B only
`MCCMSG_PRI_SIGNAL_VC_AUXILIARY1`	voltage clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_VC_AUXILIARY2`	voltage clamp, 700B only
`MCCMSG_PRI_SIGNAL_IC_MEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_MEMBCURRENT`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_CMDCURRENT`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_100XACMEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_EXTCMDCURRENT`	current clamp, 700B only
`MCCMSG_PRI_SIGNAL_IC_AUXILIARY1`	current clamp, 700B and 700A
`MCCMSG_PRI_SIGNAL_IC_AUXILIARY2`	current clamp, 700B only

Example
```
#include "AxMultiClampMsg.h"

// get the primary output
UINT uSignalID = 0;
if( !MCCMSG_GetPrimarySignal(m_hMCCmsg, &uSignalID, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPrimarySignalGain(HMCCMSG hMCCmsg, double dGain, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dGain	The primary output signal gain.
pnError	Address of error return code.

Remarks
Set the primary output signal gain. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the primary output signal gain
double dGain = 10;
if( !MCCMSG_SetPrimarySignalGain(m_hMCCmsg, dGain, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPrimarySignalGain(HMCCMSG hMCCmsg, double *pdGain, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdGain	Address of return primary output signal gain.
pnError	Address of error return code.

Remarks
Get the primary output signal gain. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the primary output signal gain
double dGain = 0;
if( !MCCMSG_GetPrimarySignalGain(m_hMCCmsg, &dGain, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPrimarySignalLPF(HMCCMSG hMCCmsg, double dLPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dLPF	The primary output signal lowpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Set the primary output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the primary output signal lowpass filter frequency
double dLPF = 1e4;  // 10 kHz
if( !MCCMSG_SetPrimarySignalLPF(m_hMCCmsg, dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPrimarySignalLPF(HMCCMSG hMCCmsg, double *pdLPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdLPF	Address of return primary output signal lowpass filter frequency in Hertz.

pnError	Address of error return code.

Remarks
Get the primary output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the primary output signal lowpass filter frequency
double dLPF = 0;
if( !MCCMSG_GetPrimarySignalLPF(m_hMCCmsg, &dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPrimarySignalHPF(HMCCMSG hMCCmsg, double dHPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dHPF	The primary output signal highpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Set the primary output signal highpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the primary output signal highpass filter frequency
double dHPF = 3e4;  // 30 kHz
if( !MCCMSG_SetPrimarySignalHPF(m_hMCCmsg, dHPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPrimarySignalHPF(HMCCMSG hMCCmsg, double *pdHPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdHPF	Address of return primary output signal highpass filter in Hertz. frequency.
pnError	Address of error return code.

Remarks
Get the primary output signal highpass filter frequency in Hertz. Supports voltage and current clamp.

Example

```
#include "AxMultiClampMsg.h"

// get the primary output signal highpass filter frequency
double dHPF = 0;
if( !MCCMSG_GetPrimarySignalHPF(m_hMCCmsg, &dHPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Scope Signal

BOOL MCCMSG_SetScopeSignalLPF(HMCCMSG hMCCmsg, double dLPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dLPF	The scope output signal lowpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Set the scope output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the scope output signal lowpass filter frequency
double dLPF = 1e4;  // 10 kHz
if( !MCCMSG_SetScopeSignalLPF(m_hMCCmsg, dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetScopeSignalLPF(HMCCMSG hMCCmsg, double *pdLPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdLPF	Address of return scope output signal lowpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Get the scope output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the scope output signal lowpass filter frequency
double dLPF = 0;
if( !MCCMSG_GetScopeSignalLPF(m_hMCCmsg, &dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Secondary (or Raw) Signal

BOOL MCCMSG_SetSecondarySignal(HMCCMSG hMCCmsg, UINT uSignalID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uSignalID	The secondary output signal.
pnError	Address of error return code.

Remarks
Set the secondary output signal. Supports voltage and current clamp. `uSignalID` must be filled in as one of the following values:

`MCCMSG_SEC_SIGNAL_VC_MEMBCURRENT`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_MEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_PIPPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_100XACMEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_EXTCMDPOTENTIAL`	voltage clamp, 700B only
`MCCMSG_SEC_SIGNAL_VC_AUXILIARY1`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_AUXILIARY2`	voltage clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_MEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_MEMBCURRENT`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_CMDCURRENT`	current clamp, 700A only
`MCCMSG_SEC_SIGNAL_IC_PIPPOTENTIAL`	current clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_100XACMEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_EXTCMDCURRENT`	current clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_AUXILIARY1`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_AUXILIARY2`	current clamp, 700B only

Example
```
#include "AxMultiClampMsg.h"

// set the secondary output to external command potential in voltage clamp
UINT uSignalID = MCCMSG_PRI_SIGNAL_VC_EXTCMDPOTENTIAL;
if( !MCCMSG_SetSecondarySignal(m_hMCCmsg, uSignalID, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSecondarySignal(HMCCMSG hMCCmsg, UINT *puSignalID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
puSignalID	Address of return secondary output signal.
pnError	Address of error return code.

Remarks
Get the secondary output signal. Supports voltage and current clamp. `uSignalID` will be filled out as one of the following values:

`MCCMSG_SEC_SIGNAL_VC_MEMBCURRENT`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_MEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_PIPPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_100XACMEMBPOTENTIAL`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_EXTCMDPOTENTIAL`	voltage clamp, 700B only
`MCCMSG_SEC_SIGNAL_VC_AUXILIARY1`	voltage clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_VC_AUXILIARY2`	voltage clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_MEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_MEMBCURRENT`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_CMDCURRENT`	current clamp, 700A only
`MCCMSG_SEC_SIGNAL_IC_PIPPOTENTIAL`	current clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_100XACMEMBPOTENTIAL`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_EXTCMDCURRENT`	current clamp, 700B only
`MCCMSG_SEC_SIGNAL_IC_AUXILIARY1`	current clamp, 700B and 700A
`MCCMSG_SEC_SIGNAL_IC_AUXILIARY2`	current clamp, 700B only

Example
```
#include "AxMultiClampMsg.h"

// get the secondary output
UINT uSignalID = 0;
if( !MCCMSG_GetSecondarySignal(m_hMCCmsg, &uSignalID, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSecondarySignalGain(HMCCMSG hMCCmsg, double dGain, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dGain	The secondary output signal gain.
pnError	Address of error return code.

Remarks
Set the secondary output signal gain. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the secondary output signal gain
double dGain = 10;
if( !MCCMSG_SetSecondarySignalGain(m_hMCCmsg, dGain, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSecondarySignalGain(HMCCMSG hMCCmsg, double *pdGain, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdGain	Address of return secondary output signal gain.
pnError	Address of error return code.

Remarks
Get the secondary output signal gain. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the secondary output signal gain
double dGain = 0;
if( !MCCMSG_GetSecondarySignalGain(m_hMCCmsg, &dGain, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetSecondarySignalLPF(HMCCMSG hMCCmsg, double dLPF, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dLPF	The secondary output signal lowpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Set the secondary output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the secondary output signal lowpass filter frequency
double dLPF = 1e4;  // 10 kHz
if( !MCCMSG_SetSecondarySignalLPF(m_hMCCmsg, dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetSecondarySignalLPF(HMCCMSG hMCCmsg, double *pdLPF, int *pnError);

Parameter	Description
hMCCmsg	Handle to message handler object.
pdLPF	Address of return secondary output signal lowpass filter frequency in Hertz.
pnError	Address of error return code.

Remarks
Get the secondary output signal lowpass filter frequency in Hertz. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the secondary output signal lowpass filter frequency
double dLPF = 0;
if( !MCCMSG_GetSecondarySignalLPF(m_hMCCmsg, &dLPF, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Output Zero

BOOL MCCMSG_SetOutputZeroEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The output zero enable state.
pnError	Address of error return code.

Remarks
Set the output zero enable state. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable output zero
BOOL bEnable = TRUE;
if( !MCCMSG_SetOutputZeroEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetOutputZeroEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return output zero enable state.
pnError	Address of error return code.

Remarks
Get the output zero enable state. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the output zero enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetOutputZeroEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetOutputZeroAmplitude(HMCCMSG hMCCmsg, double dAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dAmplitude	The output zero amplitude (Volts).
pnError	Address of error return code.

Remarks
Set the output zero amplitude in Volts. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the output zero amplitude
double dAmplitude = 0.05;  // 50 mV
if( !MCCMSG_SetOutputZeroAmplitude(m_hMCCmsg, dAmplitude, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetOutputZeroAmplitude(HMCCMSG hMCCmsg, double *pdAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdAmplitude	Address of return output zero amplitude (Volts).
pnError	Address of error return code.

Remarks
Get the output zero amplitude in Volts. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the output zero amplitude
double dAmplitude = 0;
if( !MCCMSG_GetOutputZeroAmplitude(m_hMCCmsg, &dAmplitude, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoOutputZero(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic output zero. This calculates and compensates for offset potentials at the amplifier output. A timeout value of 3 seconds is recommended for this command. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto output zero
if( !MCCMSG_AutoOutputZero(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Leak Subtraction

BOOL MCCMSG_SetLeakSubEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The leak subtraction enable state.
pnError	Address of error return code.

Remarks
Set the leak subtraction enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable leak subtraction
BOOL bEnable = TRUE;
if( !MCCMSG_SetLeakSubEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetLeakSubEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return leak subtraction enable state.
pnError	Address of error return code.

Remarks
Get the leak subtraction enable state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the leak subtraction enable state
BOOL bEnable = FALSE;
if( !MCCMSG_SetLeakSubEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetLeakSubResist(HMCCMSG hMCCmsg, double dResistance, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dResist	The leak subtraction resistance (Ohms).
pnError	Address of error return code.

Remarks
Set the leak subtraction resistance in Ohms. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the leak subtraction resistance
double dResist = 2e8; // 200 MOhms
if( !MCCMSG_SetLeakSubResist(m_hMCCmsg, dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetLeakSubResist(HMCCMSG hMCCmsg, double *pdResistance, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdResist	Address of return leak subtraction resistance (Ohms).
pnError	Address of error return code.

Remarks
Get the leak subtraction resistance in Ohms. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the leak subtraction resistance
double dResist = 0;
if( !MCCMSG_GetLeakSubResist(m_hMCCmsg, &dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoLeakSub(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic leak subtraction. A timeout value of 3 seconds is recommended for this command. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto leak subtraction
if( !MCCMSG_AutoLeakSub(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Bridge Balance

BOOL MCCMSG_SetBridgeBalEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The bridge balance enable state.
pnError	Address of error return code.

Remarks
Set the bridge balance enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable bridge balance
BOOL bEnable = TRUE;
if( !MCCMSG_SetBridgeBalEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetBridgeBalEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return bridge balance enable state.
pnError	Address of error return code.

Remarks
Get the bridge balance enable state. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the bridge balance enable state
BOOL bEnable = FALSE;
if( !MCCMSG_SetBridgeBalEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetBridgeBalResist(HMCCMSG hMCCmsg, double dResistance, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dResist	The bridge balance resistance (Ohms).
pnError	Address of error return code.

Remarks
Set the bridge balance resistance in Ohms. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the bridge balance resistance
double dResist = 1e7; // 10 MOhms
if( !MCCMSG_SetBridgeBalResist(m_hMCCmsg, dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetBridgeBalResist(HMCCMSG hMCCmsg, double *pdResistance, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdResist	Address of return bridge balance resistance (Ohms).
pnError	Address of error return code.

Remarks
Get the bridge balance resistance in Ohms. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the bridge balance resistance
double dResist = 0;
if( !MCCMSG_GetBridgeBalResist(m_hMCCmsg, &dResist, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_AutoBridgeBal(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute automatic bridge balance. A timeout value of 3 seconds is recommended for this command.
Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute auto bridge balance
if( !MCCMSG_AutoBridgeBal(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Clear

BOOL MCCMSG_ClearPlus(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute clear plus. Delivers a large positive current step down the micropipette. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute clear plus
if( !MCCMSG_ClearPlus(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_ClearMinus(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute clear minus. Delivers a large negative current step down the micropipette. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute clear minus
if( !MCCMSG_ClearMinus(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Pulse, Zap and Buzz

BOOL MCCMSG_Pulse(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute pulse. Delivers a command current or voltage down the micropipette. The pulse amplitude and duration are set by `MCCMSG_SetPulseAmplitude` and `MCCMSG_SetPulseDuration` respectively. Supports current and voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute pulse
if( !MCCMSG_Pulse(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPulseAmplitude(HMCCMSG hMCCmsg, double dAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dAmplitude	The pulse amplitude (Volts or Amps).
pnError	Address of error return code.

Remarks
Set the pulse amplitude in Volts or Amps. The pulse amplitude must be in SI units appropriate to the amplifier mode. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the pulse amplitude
double dAmplitude = 1e-1; // 100 mV
if( !MCCMSG_SetPulseAmplitude(m_hMCCmsg, dAmplitude, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPulseAmplitude(HMCCMSG hMCCmsg, double *pdAmplitude, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdAmplitude	Address of return pulse amplitude (Volts or Amps).
pnError	Address of error return code.

Remarks
Get the pulse amplitude in Volts or Amps. The return value will be in SI units appropriate to the amplifier mode. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the pulse amplitude
double dAmplitude = 0;
if( !MCCMSG_GetPulseAmplitude(m_hMCCmsg, &dAmplitude, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetPulseDuration(HMCCMSG hMCCmsg, double dDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dDuration	The pulse duration (seconds).
pnError	Address of error return code.

Remarks
Set the pulse duration in seconds. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the pulse duration
double dDuration = 1e-2; // 10 ms
if( !MCCMSG_SetPulseDuration(m_hMCCmsg, dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetPulseDuration(HMCCMSG hMCCmsg, double *pdDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.

pdDuration	Address of return pulse duration (seconds).
pnError	Address of error return code.

Remarks
Get the pulse duration in seconds. Supports voltage and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the pulse duration
double dDuration = 0;
if( !MCCMSG_GetPulseDuration(m_hMCCmsg, &dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_Zap(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute zap. Zap applies a +1 V pulse down the micropipette, used to rupture a patch when going from on-cell to whole-cell. The pulse duration is set by `MCCMSG_SetZapDuration`. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute zap
if( !MCCMSG_Zap(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetZapDuration(HMCCMSG hMCCmsg, double dDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dDuration	The zap duration (seconds).
pnError	Address of error return code.

Remarks
Set the zap duration in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"
```

```
// set the zap duration
double dDuration = 1e-2; // 10 ms
if( !MCCMSG_SetZapDuration(m_hMCCmsg, dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetZapDuration(HMCCMSG hMCCmsg, double *pdDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdDuration	Address of return zap duration (seconds).
pnError	Address of error return code.

Remarks
Get the zap duration in seconds. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the zap duration
double dDuration = 0;
if( !MCCMSG_GetZapDuration(m_hMCCmsg, &dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_Buzz(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute buzz. Buzz applies a brief, strong 10 kHz current down the micropipette, used to aid penetration of the cell membrane, and to clear blocked micropipette tips. The buzz duration is set by `MCCMSG_SetBuzzDuration`. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// execute buzz
if( !MCCMSG_Buzz(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetBuzzDuration(HMCCMSG hMCCmsg, double dDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
dDuration	The buzz duration (seconds).
pnError	Address of error return code.

Remarks
Set the buzz duration in seconds. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// set the buzz duration
double dDuration = 1e-2; // 10 ms
if( !MCCMSG_SetBuzzDuration(m_hMCCmsg, dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetBuzzDuration(HMCCMSG hMCCmsg, double *pdDuration, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pdDuration	Address of return buzz duration (seconds).
pnError	Address of error return code.

Remarks
Get the buzz duration in seconds. Supports current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the buzz duration
double dDuration = 0;
if( !MCCMSG_GetBuzzDuration(m_hMCCmsg, &dDuration, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Meters

BOOL MCCMSG_SetMeterResistEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The resistance meter enable state.
pnError	Address of error return code.

Remarks
Set the resistance meter state. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable resistance meter
BOOL bEnable = TRUE;
if( !MCCMSG_SetMeterResistEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetMeterResistEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return resistance meter enable state.
pnError	Address of error return code.

Remarks
Get the resistance meter state. Supports voltage clamp and current clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the resistance meter enable state
BOOL bEnable = FALSE;
if( !MCCMSG_GetMeterResistEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_SetMeterIrmsEnable(HMCCMSG hMCCmsg, BOOL bEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
bEnable	The Irms meter enable state.
pnError	Address of error return code.

Remarks
Set the Irms meter state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// enable Irms meter
BOOL bEnable = TRUE;
if( !MCCMSG_SetMeterIrmsEnable(m_hMCCmsg, bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_GetMeterIrmsEnable(HMCCMSG hMCCmsg, BOOL *pbEnable, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pbEnable	Address of return Irms meter enable state.
pnError	Address of error return code.

Remarks
Get the Irms meter state. Supports voltage clamp.

Example
```
#include "AxMultiClampMsg.h"

// get the Irms meter enable state
BOOL bEnable = FALSE;
if( !MCCMSG_SetMeterIrmsEnable(m_hMCCmsg, &bEnable, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Tool Bar

BOOL MCCMSG_Reset(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Execute reset. Resets all MultiClamp 700x Commander options to their default values.

Example
```
#include "AxMultiClampMsg.h"

// execute reset
if( !MCCMSG_Reset(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_ToggleAlwaysOnTop(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Toggle "Always On Top". Puts MultiClamp 700x Commander on top of other applications, even when it is not the active window.

Example
```
#include "AxMultiClampMsg.h"

// toggle "Always On Top"
if( !MCCMSG_ToggleAlwaysOnTop(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_ToggleResize(HMCCMSG hMCCmsg, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
pnError	Address of error return code.

Remarks
Toggle resize. This command resizes the MultiClamp 700B Commander interface. It toggles the interface between: full display, meter panel, and manually resized (only available if you have resized the interface by dragging the edges or corners).

Example
```
#include "AxMultiClampMsg.h"

// toggle resize
if( !MCCMSG_ToggleResize(m_hMCCmsg, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

BOOL MCCMSG_QuickSelectButton(HMCCMSG hMCCmsg, UINT uButtonID, int *pnError);

Parameter	**Description**
hMCCmsg	Handle to message handler object.
uButtonID	The Quick Select Button ID
pnError	Address of error return code.

Remarks
Execute the specified "Quick Select Button". A "Quick Select Button" will open a saved configuration (*.mcc files) of the MultiClamp 700B Commander, or start an executable file (*.exe).

Example
```
#include "AxMultiClampMsg.h"

// execute quick select button 1
UINT uButtonID = MCCMSG_QSB_1;
if( !MCCMSG_QuickSelectButton(m_hMCCmsg, uButtonID, &nError) )
{
   char szError[256] = "";
   MCCMSG_BuildErrorText(m_hMCCmsg, nError, szError, sizeof(szError));
   AfxMessageBox(szError, MB_ICONSTOP);
}
```

Errors handling

BOOL CLXMSG_BuildErrorText (HCLXMSG hClxmsg, int nErrorNum, LPSTR sTxtBuf, UINT uMaxLen);

Parameter	**Description**
hClxmsg	Handle to message handler object.
nErrorNum	The error code.
sTxtBuf	Char buffer to return error string
uMaxLen	Size of char buffer

Remarks
Return the error as a text string.

Example
```
#include "AxClampexMsg.h"

// get error as a string
char szError[256] = "";
int nError = CLXMSG_ERROR_PROTOCOLPATHNOTSET;
CLXMSG_BuildErrorText(m_hClxmsg, nError, szError, sizeof(szError));
AfxMessageBox(szError, MB_ICONSTOP);
```