

MEServo.dll is an interface that can be used to control the servo. Definitions for the interface are in MEServo.h. The import library is MEServo.lib. All functions use WINAPI (“C”) calling convention.

Error codes

Unless otherwise noted, all MEServo functions return MES_OK (0) on success, or one of the following error codes:

MES_OK	successful call
MES_CANCEL	Cancel button from dialog
MES_ERROR_INIT_FAIL	DLL initialization failure or MES_Initialize not called
MES_ERROR_EXCEPTION	unhandled exception occurred
MES_ERROR_BUSY	call not allowed due to tape motion, etc.
MES_ERROR_OPEN_FAIL	could not access serial port (config error?)
MES_ERROR_NO_SERVO	no response from servo (wrong serial port?)
MES_ERROR_NOT_OPEN	Serial port is not open
MES_ERROR_INVALID_FILE	bad file or path for inifile
MES_ERROR_INVALID_PARAM	bad value in parameter struct (check dwSize!)
MES_ERROR_PARSE	response from servo could not be parsed
MES_ERROR_MENU	couldn't change menus (servo crashed?)
MES_ERROR_UNSUPPORTED	command not supported on this firmware version
MES_ERROR_TIMEOUT	servo has stopped responding (can be caused by severely out-of-sync menu state.)

Initialization Functions

DWORD MES_GetInterfaceVersion ()

Returns the interface version. The high word of the returned value is the major version number and low word is the minor version number. Changes in minor version number do not affect the structure of the interface; changes to the interface will be indicated by stepping the major version number. This documentation is for version 1.x.

The interface version can be displayed as the DLL version; the minor version number should be interpreted as separate minor version and build numbers; i.e., major version 1, minor version 203 (decimal) should be displayed as “1.2.3” or “1.2 build 3”.

DWORD MES_Initialize (HWND hwndOwner, LPCSTR lpszInifileName = NULL)

Initializes the DLL. `hwndOwner` is the window handle for the application’s main window. `lpszInifileName` is the inifile that MEServo.DLL will use to store its settings; it must be a fully qualified file name. If `lpszInifileName` is NULL, “MEServoDLL.ini” in the calling application’s directory will be used.

Note: you must call MES_Terminate before `hwndOwner` is destroyed.

DWORD MES_Terminate ()

Cleans up resources used by the DLL. Must be called before the window passed into MES_Initialize is destroyed.

DWORD MES_RegisterCallback (tMESCallbackProc * pStatusCallback)

Registers a function that will be called with status information during tape operation. The function must be declared as:

```
void WINAPI StatusCallback (DWORD dwType, DWORD dwData)
```

Call RegisterCallback(NULL) to unregister the callback.

The actual type of `dwData` depends on `dwType` as follows:

<u>dwType</u>	<u>dwData</u>
MES_RESET	unused
MES_VERSION	DWORD: firmware version
MES_BOT	unused
MES_EOT	unused
MES_DONE	unused
MES_PASS	DWORD: pass number
MES_STATUS	LPVOID: pointer to tMESStatus
MES_SERVO	LPVOID: pointer to tMESServoStatus
MES_ANALOG	LPVOID: pointer to tMESAnalogInputs
MES_ERROR	LPVOID: pointer to tMESError
MES_DATA_VISIBLE	bool: data window visible

You must copy the data that you need from the structures; they are temporary objects.

Serial Port Functions

DWORD MES_LoadConfig (LPCSTR lpszInifileName = NULL)

Loads serial port configuration information from an inifile. The serial port must be closed to make this call. `lpszInifileName` must be a fully qualified file name. If `lpszInifileName` is NULL the inifile passed into MES_Initialize will be used.

Returns MES_OK (0) or an error value.

DWORD MES_StoreConfig (LPCSTR lpszInifileName = NULL)

Stores serial port configuration information to an inifile. `lpszInifileName` must be a fully qualified file name. If `lpszInifileName` is NULL the inifile passed into `MES_Initialize` will be used.

Returns `MES_OK` (0) or an error value.

DWORD MES_EditConfig ()

Brings up a dialog to allow the serial port setting to be changed. The serial port must be closed to make this call.

Returns `MES_OK` (0) or an error value. In particular, expect `MES_ERROR_BUSY` if the comm port is open.

Note: when `MES_EditConfig` returns `MES_OK` you normally want to call `MES_StoreConfig` to save the new value.

DWORD MES_Open ()

Opens the serial port and initiates communication with the servo controller. The default serial port settings are COM1, 57600 bps, 8 bits, no parity, 1 stop bit, software handshaking.

Returns `MES_OK` (0) if the port was opened, or an error value..

Note: `MES_Open` does not wait for the servo controller to respond. You must detect `MES_VERSION` in your status callback to know that the servo controller has responded.

The serial port settings can be changed by calling `MES_LoadConfig` or `MES_EditConfig` before calling `MES_Open`.

DWORD MES_Close ()

Closes the serial port. Note that the servo controller is not affected by closing the serial port; it will continue any operation that is in progress. You may want to call `MES_Abort` before calling `MES_Close` to ensure that the servo controller is stopped.

Returns `MES_OK` (0) or an error value.

Status and Configuration Functions

DWORD MES_GetFirmwareVersion ()

Returns the firmware version. The high word of the returned value is the major version number and low word is the minor version number. In early versions of the firmware, it is not always possible to get the firmware version, in which case the return value will be 0; assume firmware version lower than 3.11 in this case. *If the high-order bit of the return value is set, an error has occurred. See the “Error Values” section of this document.*

DWORD MES_GetStatus (tMESStatus * pStatus)
DWORD MES_RequestStatus ()

Returns information about the Servo Controller. pStatus is a pointer to a tMESStatus structure:

```
struct tMESStatus {
    DWORD    dwSize;           // must be sizeof (tMESStatus)
    DWORD    dwStatus;        // status bits -- see tMESStatusBits
    LONG     dwSupplyPos;     // supply motor encoder position
    LONG     dwTakeupPos;     // takeup motor encoder position
    DWORD    dwTension1;     // tension sensor 1 in mN
    DWORD    dwTension2;     // tension sensor 2 in mN
};
```

MES_RequestStatus issues the request to the servo and returns immediately. The requested information will be returned via the Status Callback function.

These functions can be called during tape motion but there are limitations depending on firmware version; expect to get MES_ERROR_BUSY.

DWORD MES_GetStatusBits ()

This function returns the servo status bits as defined in tMESStatusBits:

```
enum tMESStatusBits {
    MES_TENSION           = 0x00000001,
    MES_MOTION           = 0x00000002,
    MES_DIRECTION        = 0x00000004,
    MES_SLOW             = 0x00000008,
    MES_WIND             = 0x00000010,
    MES_GOTO_POS        = 0x00000020,
    MES_WEAR            = 0x00000040,
    MES_UNLOAD          = 0x00000080,
    MES_OFF             = 0x00000000,
    MES_STOP           = MES_TENSION,
    MES_FORWARD        = MES_TENSION | MES_MOTION,
    MES_REVERSE       = MES_FORWARD | MES_DIRECTION,
    MES_SLOW_FORWARD  = MES_SLOW | MES_FORWARD,
    MES_SLOW_REVERSE  = MES_SLOW | MES_REVERSE,
    MES_WIND_FORWARD  = MES_WIND | MES_FORWARD,
    MES_WIND_REVERSE  = MES_WIND | MES_REVERSE,
    MES_GOTO         = MES_GOTO_POS | MES_TENSION | MES_MOTION,
    MES_SHORT_WEAR    = MES_WEAR | MES_TENSION | MES_MOTION,
    MES_LONG_WEAR     = MES_WEAR | MES_SLOW,
    MES_UNLOAD_FORWARD = MES_UNLOAD | MES_SLOW_FORWARD,
    MES_UNLOAD_REVERSE = MES_UNLOAD | MES_REVERSE,
    MES_UNLOAD_WIND_REV = MES_UNLOAD | MES_WIND_REVERSE
};
```

DWORD MES_GetServoParams (tMESServoParams * pServoParams)
DWORD MES_SetServoParams (tMESServoParams * pServoParams)

These functions are used to modify the tension and speed parameters for the servo controller. pServoParams is a pointer to a tMESServoParams structure:

```
struct tMESServoParams {
    DWORD    dwSize;        // must be sizeof (MEServoParams)
    DWORD    dwSpeed;       // Tape speed in cm/sec
    DWORD    dwTension;     // Tape tension in mN
    CHAR     cSensor;       // '1' '2' 'B' or 'N'
};
```

For further information on these parameters, see [1]. The tape motors must be off to change the tension values; they must be stopped or off to change the speed value. To change the speed value only, set dwTension to 0 when calling MES_SetServoParams.

Returns MES_OK (0) or an error value.

DWORD MES_GetWearTestParams (tMESWearTestParams * pWearTestParams)
DWORD MES_SetWearTestParams (tMESWearTestParams * pWearTestParams)

Get or Set parameters for the Wear Tests. pWearTestParams is a pointer to a tMESWearTestParams structure:

```
struct tMESWearTestParams {
    DWORD    dwSize;        // must be sizeof (tMESWearTestParams)
    LONG     lBotLimit;     // BOT limit for test
    LONG     lEotLimit;     // EOT limit for test
    DWORD    dwCycles;      // number of cycles for the test
    LONG     lDriftRate;    // Position drift for each cycle
};
```

lDriftRate is only used of the Short Cycle Wear Test. See [1] for more info.

Servo Command Functions

DWORD MES_Abort ()

Immediately aborts tape motion and powers down the motors. This function can be called at any time.

Note: this is the only way to release tape tension.

DWORD MES_Tension ()

Applies tension to the tape. If the tape has already been tensioned or is in motion this is a no-op.

Returns MES_OK (0) or an error value.

DWORD MES_Forward ()
DWORD MES_Reverse ()
DWORD MES_SlowForward ()
DWORD MES_SlowReverse ()
DWORD MES_WindForward ()
DWORD MES_WindReverse ()

These functions start tape motion. They all return MES_OK (0) or an error value. In particular, expect MES_ERROR_BUSY.

MES_Forward, MES_Reverse, MES_SlowForward and MES_SlowReverse stop at beginning and end of tape. The status callback function will receive MES_BOT or MES_EOT. (See MES_RegisterCallback function.)

MES_WindForward and MES_WindReverse will spool off the tape with no error indication.

DWORD MES_Stop ()

Stops tape motion and maintains tension.

Returns MES_OK (0) or an error value. In particular, expect MES_ERROR_BUSY if a wear test is running.

DWORD MES_ZeroEncoders ()

Set the motor encoders to zero. This can be called while the tape is moving

DWORD MES_Goto (LONG lPos)

Start tape motion towards the specified motor encoder position. The status callback function will receive MES_DONE, MES_BOT or MES_EOT. (See MES_RegisterCallback function.)

Returns MES_OK (0) or an error value. In particular, expect MES_ERROR_BUSY.

DWORD MES_Unload ()

Unloads the tape by rewinding to BOT and then winding the supply reel. *There is no indication when unloading is done.* This function is not available in earlier versions of the firmware.

DWORD MES_LongCycleWearTest ()
DWORD MES_ShortCycleWearTest ()

These functions start the Wear Tests. They return MES_OK (0) or an error value. In particular, expect MES_ERROR_BUSY. During the Wear Tests, the status callback function will receive MES_PASS, MES_DONE, MES_BOT and MES_EOT. (See MES_RegisterCallback function.)

Data Window Commands

DWORD MES_ShowSerialData ()
DWORD MES_HideSerialData ()
DWORD MES_SerialDataIsVisible ()

MEServo.DLL can display the data it communicates with the servo controller. These functions control this data window.

DWORD MES_EnableKeyboard ()
DWORD MES_DisableKeyboard ()

These functions enable the keyboard in the data window. Normally the keyboard is disabled except for <esc> when the data window has focus.

References

[1] Griffiths, Jonathan. “Servo Console Operating Instructions.” 18 March 2002. Mountain Engineering II, Inc. 20 June 2003. <http://mountainengineering.com/mts/servo_console.pdf >