

MH1034 Mono DLL Description

Document History:

Author	Revision	Date	Revised Page & Paragraph	Remarks	Approval Date
B. Bonnett	1.0	July 28, 2016	All	Initial Draft	NA
Jay Martin	1.1	April 6, 2018	All	Added Remote Scan functions for triggered scanning	NA
Jay Martin	1.2	April 29, 2021		Documented Updated enumeration method for MH1034, multiple devices Remove: HJYFindMH1034Mono Add: HJYGetAvailableUnits	
Jay Martin	1.3	June 1, 2024		Added documentation regarding requirement to call HJYInitializeLibrary() prior to other calls.	

This document describes the interface to the MH1034 DLL and its supported functionality.

Contents

Document History:	1
Supported MH1034 DLL Functions.....	3
General Overview	3
HJYInitializeLibrary()	4
HJYCloseLibrary()	4
HJYGetAvailableUnits(int *count, char *buff, int buffSize).....	4
HJYInitializeMono().....	4
HJYReadInitStatus()	4
HJYMotorBusy()	5
HJYReadCurrentWavelength()	5
HJYMoveToWavelengthWaitBusy().....	5

HJYMoveToWavelengthNoBusy()	6
HJYReadCurrentStepPosition()	6
HJYMoveToStepPosition()	6
HJYFwReadCurrentPosition()	6
HJYFwMoveToPositionWaitBusy()	7
HJYFwMoveToPositionNoBusy()	7
HJYSetScanParams()	7
HJYStartScan()	8
HJYStopScan()	8
HJYGetNumScanWavelengths()	8
HJYReadScanWavelength()	8
HJYClearScanParams()	9
HJYGetFirstScanParam()	9
HJYGetNextParam()	9
HJYScanAddWavelength()	9
HJYScanSetFWTransition()	10
HJYScanGetFWTransition()	10
HJYScanGetFWLowPosition()	10
HJYScanGetWavelengths()	11
HJYGetWavelengthBoundaries()	11
HJYGetScanMaxWavelengthEntries()	11
HJYDeviceHasTriggers()	11

Supported MH1034 DLL Functions

```
WIN32DLL_API HJYInitializeLibrary();
WIN32DLL_API HJYCloseLibrary();
WIN32DLL_API HJYGetAvailableUnits(int *count, char *buff, int buffSize);
WIN32DLL_API HJYInitializeMono();
WIN32DLL_API HJYReadInitStatus(long* initStatus);
WIN32DLL_API HJYMotorBusy(BOOL* isBusy);
WIN32DLL_API HJYReadCurrentWavelength(float* wl);
WIN32DLL_API HJYMoveToWavelengthWaitBusy(float wl);
WIN32DLL_API HJYMoveToWavelengthNoBusy(float wl);
WIN32DLL_API HJYReadCurrentStepPosition(long* stepPos);
WIN32DLL_API HJYMoveToStepPosition(long stepPos);
WIN32DLL_API HJYFwReadCurrentPosition(long* position);
WIN32DLL_API HJYFwMoveToPositionWaitBusy(long position);
WIN32DLL_API HJYFwMoveToPositionNoBusy(long position);
WIN32DLL_API HJYSetScanParams( float fStartWL, float fWLStepSize, float fEndWL , const char * strSerialNum= '\0');
WIN32DLL_API HJYStartScan( const char * strSerialNum= '\0');
WIN32DLL_API HJYStopScan( const char * strSerialNum= '\0');
WIN32DLL_API HJYGetNumScanWavelengths( WORD * numScanWLs, const char * strSerialNum= '\0');
WIN32DLL_API HJYReadScanWavelength( WORD *wCurrWLNum, const char * strSerialNum= '\0');
WIN32DLL_API HJYClearScanParams( const char * strSerialNum= '\0');
WIN32DLL_API HJYGetFirstScanParam( float *param , const char * strSerialNum= '\0');
WIN32DLL_API HJYGetNextParam( float *param , const char * strSerialNum= '\0');
WIN32DLL_API HJYScanAddWavelength( float *wl,const char * strSerialNum= '\0' );
WIN32DLL_API HJYScanSetFWTransition( float fTransitionWL, WORD iLowWLFilterPos, const char * strSerialNum= '\0');
WIN32DLL_API HJYScanGetFWTransition( float *fTransitionWL, const char * strSerialNum= '\0');
WIN32DLL_API HJYScanGetFWLowPosition( WORD *wLowFWPos, const char * strSerialNum= '\0');
WIN32DLL_API HJYScanGetWavelengths( WORD howMany, float* wlValues, const char * strSerialNum= '\0');
WIN32DLL_API HJYGetWavelengthBoundaries( float * minWL, float* maxWL, const char * strSerialNum= '\0');
WIN32DLL_API HJYGetScanMaxWavelengthEntries( WORD *maxNumEntries, const char * strSerialNum= '\0');
WIN32DLL_API HJYDeviceHasTriggers( int *hasTriggers, const char * strSerialNum= '\0');
```

General Overview

This library provides support for the MH1034 Spectrometer. Currently, there are 2 models with the only difference being one has Triggers capabilities and the other does not.

You can determine which you have by calling HJYDeviceHasTriggers() function detailed below.

If you only have a single device connected at any given time, you can omit the strSerialNumber parameter from your calls and the library will assume the first device found. Otherwise, you must send the serial number parameter in order for the library to identify which device your command is targeting.

The scanning operations of the device assume an input trigger to move to the next position in the scan table.

HJYInitializeLibrary()

Function: (BOOL) WIN32DLL_API HJYInitializeLibrary()

Description: Performs necessary initialization and checks for connected devices.

Parameters: None

Returns: TRUE – if successful, FALSE – failure on initialization

NOTE: This should be the first call made into this DLL.

HJYCloseLibrary()

Function: (BOOL) WIN32DLL_API HJYCloseLibrary()

Description: Cleans up and closes any open connections to devices. Should be called on application exit.

Parameters: None

Returns: TRUE – if successful, FALSE – failure

HJYGetAvailableUnits(int *count, char *buff, int buffSize)

Function: (BOOL) WIN32DLL_API HJYGetAvailableUnits(int *count, char *buff, int buffSize)

Description: Searches for an attached MH1034 device

Parameters: count – will contain the number of units found

buff – will contain the comma delimited list of serial numbers to be used to address the specific device

buffsize – size of the string buffer to hold the resulting list

Returns: INT. Returns the count of units found. 0 if no units are detected

NOTE: Must call HJYInitializeLibrary prior to **any** other function calls

HJYInitializeMono()

Function: (BOOL) WIN32DLL_API HJYInitializeMono(const char * strSerialNum);

Description: Initializes the Mono device hardware and returns when initialization is complete.

Parameters: None

Returns: BOOL. Returns TRUE if the Mono initialized successfully and Mono status is ready.

HJYReadInitStatus()

Function: (BOOL) WIN32DLL_API HJYReadInitStatus([out] long initStatus, const char * strSerialNum);

Description: Returns the initialization status of the Mono device hardware.

Parameters: [out] long initStatus

initStatus value	Status
0x00000000	Initialized
0x00000010	Initializing
0x00000011	Mono motor is busy
0x00000012	Filter Wheel motor is busy
0x000000E0	Un-initialized
0x000000E1	Waiting for Mono motor to stop after receiving Stop command
0x000000E2	Waiting for Filter Wheel motor to stop after receiving Stop command

Returns: BOOL. Returns TRUE if the function was successful.

HJYMotorBusy()

Function: (BOOL) WIN32DLL_API HJYMotorBusy(BOOL* isBusy, const char * strSerialNum);

Description: Returns the Busy state of the Mono

Parameters: [out] BOOL* isBusy (TRUE = Busy; FALSE = Not Busy)

Returns: BOOL. Returns TRUE if the function was successful.

HJYReadCurrentWavelength()

Function: (BOOL) WIN32DLL_API HJYReadCurrentWavelength(float* wavelength, const char * strSerialNum);

Description: Returns the current Mono wavelength position in nanometers.

Parameters: [out] float* wavelength

Returns: BOOL. Returns TRUE if the function was successful.

HJYMoveToWavelengthWaitBusy()

Function: (BOOL) WIN32DLL_API HJYMoveToWavelengthWaitBusy(float wavelength, const char * strSerialNum);

Description: Moves the Mono to a nanometer wavelength position.

The function call sends the Move command to the hardware and will continuously check the status waiting for the move to complete. When the status is no longer busy, the function call will return.

Parameters: [in] float wavelength

Returns: BOOL. Returns TRUE if the function was successful.

HJYMoveToWavelengthNoBusy()

Function: (BOOL) WIN32DLL_API HJYMoveToWavelengthNoBusy(float wavelength, const char * strSerialNum);

Description: Moves the Mono to a nanometer wavelength position.

The function call sends the Move command to the hardware and immediately returns without checking the Busy status. The calling application should then implement its own status check using HJYMotorBusy() to determine when the Mono is no longer busy.

This function maybe useful when used with a Filter Wheel move. Both Mono and Filter Wheel moves may be sent allowing both to move simultaneously, instead of doing each move separately. The calling application would then only need to wait until Busy is no longer active.

Parameters: [in] float wavelength

Returns: BOOL. Returns TRUE if the function was successful.

HJYReadCurrentStepPosition()

Function: (BOOL) WIN32DLL_API HJYReadCurrentStepPosition(long* stepPos, const char * strSerialNum);

Description: Returns the current Mono step motor position in steps.

Parameters: [out] long* stepPos

Returns: BOOL. Returns TRUE if the function was successful.

HJYMoveToStepPosition()

Function: (BOOL) WIN32DLL_API HJYMoveToStepPosition(long stepPos, const char * strSerialNum);

Description: Returns the current Mono step motor position in steps.

Description: Moves the Mono motor to step position.

The function call sends the Move command to the hardware and immediately returns without checking the Busy status. The calling application should then implement its own status check using HJYMotorBusy() to determine when the Mono is no longer busy.

Parameters: [in] long stepPos

Returns: BOOL. Returns TRUE if the function was successful.

HJYFwReadCurrentPosition()

Function: (BOOL) WIN32DLL_API HJYFwReadCurrentPosition(long* position, const char * strSerialNum);

Description: Returns the current Filter Wheel position.

Parameters: [out] long* position

Returns: BOOL. Returns TRUE if the function was successful.

HJYFwMoveToPositionWaitBusy()

Function: (BOOL) WIN32DLL_API HJYFwMoveToPositionWaitBusy(long position, const char * strSerialNum);

Description: Moves the Filter Wheel to a position.

The function call sends the Move command to the hardware and will continuously check the status waiting for the move to complete. When the status is no longer busy, the function call will return.

Parameters: [in] long position

Returns: BOOL. Returns TRUE if the function was successful.

HJYFwMoveToPositionNoBusy()

Function: (BOOL) WIN32DLL_API HJYFwMoveToPositionWaitBusy(long position, const char * strSerialNum);

Description: Moves the Filter Wheel to a position.

The function call sends the Move command to the hardware and immediately returns without checking the Busy status. The calling application should then implement its own status check using HJYMotorBusy() to determine when the Filter Wheel is no longer busy.

This function maybe useful when used with a Mono move. Both Mono and Filter Wheel moves may be sent allowing both to move simultaneously, instead of doing each move separately. The calling application would then only need to wait until Busy is no longer active.

Parameters: [in] long position

Returns: BOOL. Returns TRUE if the function was successful.

HJYSetScanParams()

Function: WIN32DLL_API HJSetScanParams(float fStartWL, float fWLStepSize, float fEndWL , const char * strSerialNum= '\0');

Description: Sets the remote scanning parameters

NOTE: Scan created is limited to number of entries returned by HJYGetScanMaxWavelengthEntries. Call will fail if the specified parameters create a table larger than this limit.

Parameters: [in] float fStartWL – Starting wavelength for the scan
[in] float fEndWL – Ending wavelength for the scan

[in] float fWLStepSize – wavelength increment between positions

Returns: BOOL. Returns TRUE if the function was successful.

HJYStartScan()

Function: WIN32DLL_API HJYStartScan(const char * strSerialNum);

Description: Starts the scan based on the previously set parameters

Parameters: [in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYStopScan()

Function: WIN32DLL_API HJYStopScan(const char * strSerialNum= '\0');

Description: Stops the currently active scan

Parameters: [in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYGetNumScanWavelengths()

Function: WIN32DLL_API HJYGetNumScanWavelengths(WORD * numScanWLs, const char * strSerialNum= '\0');

Description: Retrieves the number of entries in the currently configured scan table

Parameters: [out] WORD numScanWLs – number of entries in current scan table
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYReadScanWavelength()

Function: WIN32DLL_API HJYReadScanWavelength(WORD *wCurrWLNum, const char * strSerialNum= '\0');

Description: During a scan, returns the index in the scan table of the current position

Parameters: [out] WORD wCurrWNum – index of active scan into the scan table
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYClearScanParams()

Function: WIN32DLL_API HJYClearScanParams(const char * strSerialNum= '\0');

Description: Clears the scan table of any configured positions

Parameters: [in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYGetFirstScanParam()

Function: WIN32DLL_API HJYGetFirstScanParam(float *param , const char * strSerialNum= '\0');

Description: Gets the first entry (wavelength) in the scan table. A value of -1 indicates end of the list

Parameters: [out] param – wavelength value for the first scan table position
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYGetNextParam()

Function: WIN32DLL_API HJYGetNextParam(float *param , const char * strSerialNum= '\0');

Description: Gets the next entry (wavelength) in the scan table. A value of -1 indicates end of the list
Must call HJYGetFirstParam before calling this function

Parameters: [out] param – wavelength value for the scan table position based on the enumeration
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYScanAddWavelength()

Function: WIN32DLL_API HJYScanAddWavelength(float wl, const char * strSerialNum= '\0');

Description: Adds a single wavelength to the end of the current scan table

Parameters: [in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYScanSetFWTransition()

Function: WIN32DLL_API HJYScanSetFWTransition(float fTransitionWL, WORD iLowWLFILTERPos, const char * strSerialNum= '\0');

Description: Sets the wavelength and filter wheel position to be used as a transition point in any advanced scanning operations.

Parameters: [in] fTransitionWL
[in] iLowWLFILTERPos
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYScanGetFWTransition()

Function: WIN32DLL_API HJYScanGetFWTransition(float *fTransitionWL, const char * strSerialNum= '\0');

Description: Gets the wavelength at which the filter wheel will be placed at the specified position

Parameters: [out] fTransitionWL – wavenlength where fw will position itself based on specified settings
[in] const char * strSerialNum – Serial Number of the device you are connected to
[

Returns: BOOL. Returns TRUE if the function was successful.

HJYScanGetFWLowPosition()

Function: WIN32DLL_API HJYScanGetFWLowPosition(WORD *wLowFWPos, const char * strSerialNum= '\0');

Description: Reads position Filter wheel will be in for Low wavelengths

Parameters: [in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYScanGetWavelengths()

Function: WIN32DLL_API HJYScanGetWavelengths(WORD howMany, float* wlValues, const char * strSerialNum);

Description: Reads the specified number of entries from the current scan table

Parameters: [in] WORD howMany – number of entries to be read
[in/out] float[]: array of floats of size (howMany)
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYGetWavelengthBoundaries()

Function: WIN32DLL_API HJYGetWavelengthBoundaries(float * minWl, float* maxWl, const char * strSerialNum= '\0');

Description: Retrieve the minimum and maximum available wavelength from the device

Parameters: [out] float minWl – minimum wavelength possible for scanning
[out] float maxWl – maximum wavelength possible for scanning
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYGetScanMaxWavelengthEntries()

Function: WIN32DLL_API HJYGetScanMaxWavelengthEntries(WORD *maxNumEntries, const char * strSerialNum= '\0');

Description: Retrieves what the maximum limit is for number of entries in the scanning table

Parameters: [out] WORD maxNumEntries – size limit of entries in any constructed scan table
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.

HJYDeviceHasTriggers()

Function: WIN32DLL_API HJYDeviceHasTriggers(int *hasTriggers, const char * strSerialNum= '\0');

Description: Based on the connected product, determines if the triggering option needed for remote scanning is available.

Parameters: [out] int hasTriggers – 0 == no triggers, 1 == triggers
[in] const char * strSerialNum – Serial Number of the device you are connected to

Returns: BOOL. Returns TRUE if the function was successful.