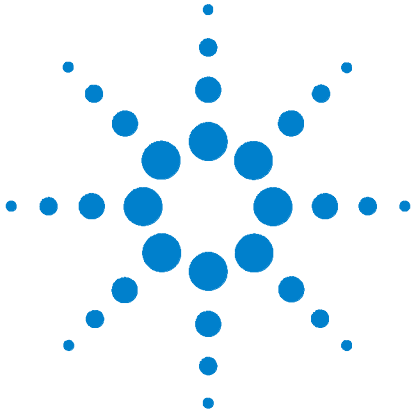


Agilent
E1735A Win32 API DLL

User's Guide

Agilent Technologies

Notices

Manual Part Number

E1735-90007

Edition

First Release February 17, 2009

Printed in USA

Agilent Technologies, Inc.
5301 Stevens Creek Boulevard
Santa Clara, CA 95051 USA

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Alphabetical List of Functions

E1735A.dll

Agilent E1735A Win32 API DLL (E1735A.dll)

External Reference Specification (ERS)

This document contains complete information needed to use the Win32 E1735A API DLL (E1735A.dll version 1.1.0.2), which is required by Windows programmers to access the E1735A USB Axis Module. This DLL provides programmer-friendly APIs that are appropriate for the Win32 programming environment. Contained herein are descriptions of all constants and function calls needed to make measurements with the E1735A device.

About the Agilent E1735A USB Axis Module

The file E1735A.dll is a DLL (dynamic link library) that offers an API (application programming interface) to the Agilent E1735A USB Axis Module. Programmers can access the E1735A via functions contained in the DLL. The E1735A is register-based. Its basic structure is shown as below.

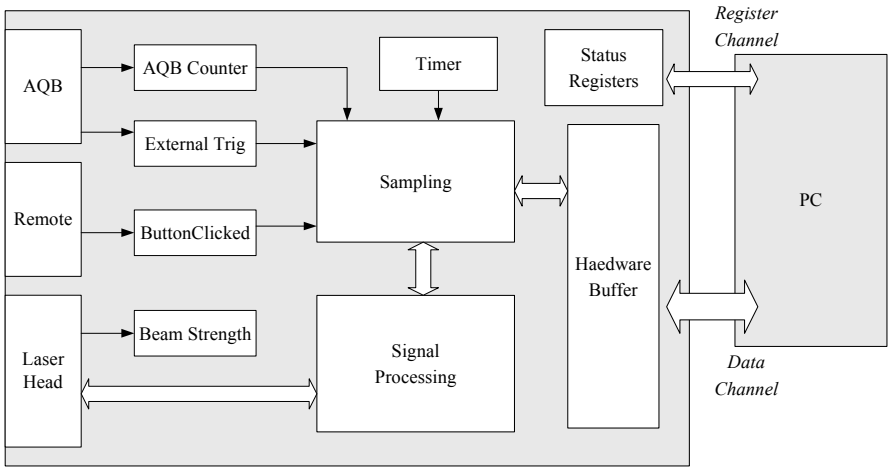


Fig.1 Diagram of the E1735A

A user can set up the E1735A and read data from it through two channels, the data channel and the register channel. Through the data channel, the user can read laser sample and related information. Using the register channel, the user can read:

- Number of samples in buffer
- Beam strength
- Button clicking
- Value of AQB counter

or send commands to:

- Start, stop and set the interval of the inner timer
- Start or stop external sampling
- Setup modulo counter, set Hysteresis value and reset AQB counter

Software and Hardware of E1735A

1. DLL Overview

There are 28 total functions in this DLL. They can be grouped as shown in Table 1 below. For more information, please refer to the section “E1735A API DLL Summary”.
The installation program places both this DLL and a support DLL, E1735ACore.dll, in the Windows system32 subdirectory. If custom installs are done, make sure both of these DLLs are placed together in a directory.

Table 1. Function list of E1735A.dll

Name	Description
• **System Information and Device Control**	
E1735A_ReadDeviceCount	Get number of available E1735A devices
E1735A_SelectDevice	Select an active device for further accessing
E1735A_GetAllRevisions	Return revision information of active device
E1735A_BlinkLED	Blink “R/W” LED
E1735A_ResetDevice	Reset laser position, clear buffers and error flags
E1735A_ReadLastError	Get the error code of last operation
• **Reading Samples**	
E1735A_ReadSampleCount	Get number of available samples that can be read
E1735A_ReadSample	Read one laser position
E1735A_ReadLastTrigger	Get the trigger of the sample that was just read
E1735A_ReadLastTimeStamp	Get the time stamp of the sample that was just read
E1735A_ReadAllSamples	Copy all captured samples to user’s buffer
E1735A_SetSampleTriggers	Enable/disable sample collecting of referred triggers
E1735A_GetSampleTriggers	Read back the setting of E1735A_SetSampleTriggers
• **Time Base Sampling**	
E1735A_SetupTimer	Set interval of timer
E1735A_StartTimer	Run timer, samples are automatically trigged by timer
E1735A_StopTimer	Stop timer
E1735A_ReadTimerSamples	Get timer-trigged samples directly
• **AQB Sampling**	
E1735A_SetupAQB	Set AQB sampling, mode, modulo and hysteresis
E1735A_ReadAQB	Get current value of AQB counter
E1735A_ReadSampleAndAQB	Get laser sample and AQB count simultaneously
• **External Sampling**	
E1735A_StartExternalSampling	Enable sampling of external trigger signal
E1735A_StopExternalSampling	Disable sampling of external trigger signal
• **Reading Status**	
E1735A_ReadButtonClicked	Get if record and/or reset button on remote control is clicked
E1735A_ReadBeamStrength	Get strength of returning beam
• **Configuring Optical Parameters**	
E1735A_SetOptics	Set type of interferometer
E1735A_GetOptics	Read back setting of E1735A_SetOptics
E1735A_SetParameter	Set parameters of environment and interferometer
E1735A_GetParameter	Read back setting of E1735A_SetParameter

2. Functions and hardware

The hardware buffer size of the E1735A is 255. To improve the performance, a software buffer is setup to extend its capability. The size of software buffer is 1048576 (2^20).
When a sample is read from hardware buffer, its trigger is checked. Only trigger types specified in the last *E1735A_SetSampleTriggers* call are allowed to pass into the software buffer. This is done via “Trigger Control”. By default, all triggers are allowed when the DLL starts.

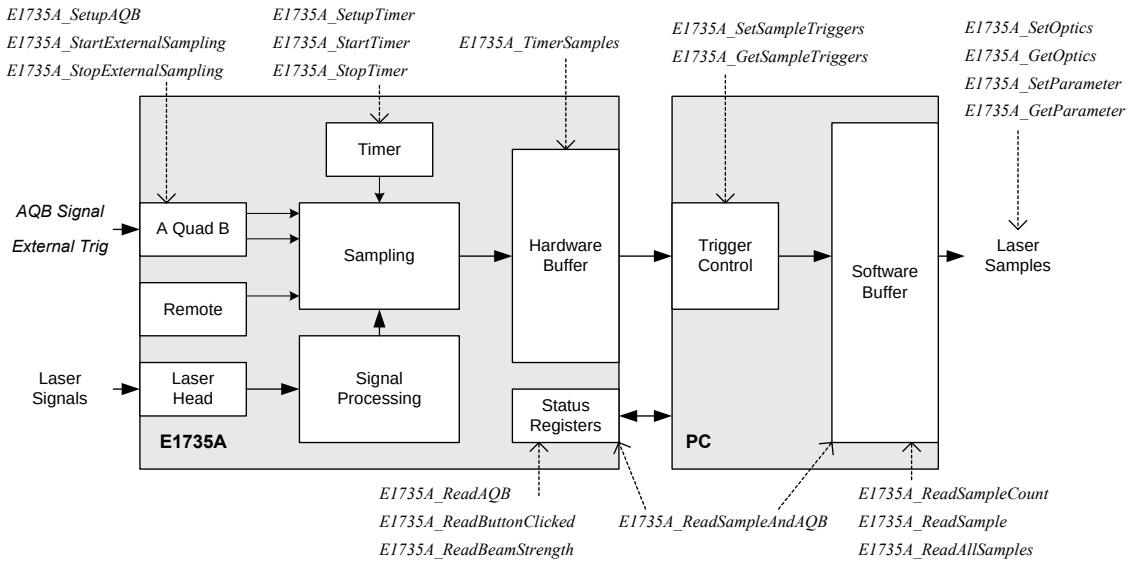


Fig.2 Relationship between E1735A and main functions of DLL

Signals from the laser head are received, processed and counted inside the E1735A. Once a trigger is active, a laser sample is latched with related information, including sampling time, trigger and error flags. This sample is then saved into the hardware buffer. Some functions can be called to configure, enable or disable these triggers. The remote record button is always enabled. Software sampling is automatically trigged only if there is no sample in either buffer, the timer is off, and a sample is required by calling the function E1735A_ReadSample.

3. Hardware working status

There are four LEDs on left side of the E1735A that indicate its working status. If the E1735A is connected but cannot be accessed through functions of the DLL, please check the “COM” LED to see if the driver has been installed correctly. When the E1735A is being initialized by DLL, “RDY” LED will blink for a few seconds and turn on afterwards.

Table 2. LEDs of the E1735A

LED label	ON	OFF
COM	Driver installed properly	Driver not installed
H.S.	USB 2.0 connection	USB 1.1 or lower
RDY	E1735A has been properly initialized	E1735A is not initialized
R/W	Reading or writing	Idle

Operating Scenarios

1. Initializing

The E1735A is automatically initialized when the E1735A.dll is loaded. When it is connected to the computer and initialized for the first time, a popup window appears to display the progress of initialization. Unless it is unplugged and connected again, this window will not appear again if the DLL is reloaded.
E1735A_ResetDevice can be used to reset the laser position, clear all buffers and reset all error flags.
If necessary, *E1735A_GetAllRevisions* can be called to make sure that the revisions of hardware and software are up to date.

2. Selecting active device

All connected E1735A's are indexed starting at 0. Use *E1735A_ReadDeviceCount* to get the number of connected devices. The first device is active by default.
The function *E1735A_SelectDevice* should be called to switch to another device. There can be only one active device at a time.
The function *E1735A_BlinkLED* can be used to identify the selected device visually.

3. Reading a sample

There are five triggers that can initiate laser samples. They are shown as below.

- Software command
- Timer
- Overflow or underflow of AQB counter
- Remote record button
- External trigger signal

No matter how the sample is initiated, the data is stored sequentially in a buffer. To get a laser sample, the user can simply call *E1735A_ReadSample*. One laser sample will always be returned regardless of how it is trigged and where it is buffered, unless the software trigger is disabled by *E1735A_SetSampleTriggers*.
E1735A_ReadLastTrigger, *E1735A_ReadLastTimeStamp* and *E1735A_ReadLastError* should be called immediately after *E1735A_ReadSample* to get information related to the last sample.
The return value of *E1735A_ReadSample* depends on the interferometer, environment and unit. See the description of following functions for more information:
E1735A_SetOptics, E1735A_GetOptics, E1735A_SetParameter, E1735A_GetParameter

4. Getting samples with AQB, external trigger or timer

Before reading samples trigged by AQB, external trigger or timer, make sure that the responding trigger is enabled by calling *E1735A_SetSampleTriggers*.
Call *E1735A_ReadSampleCount* to get the number of all samples in both the hardware buffer and the software buffer. Samples in the hardware buffer are moved to the software buffer by this function.
Next, call *E1735A_ReadAllSamples* to read out all sample data. To avoid any confusion and potential conflict that could be caused by different triggers, the user should enable only one type of trigger at a time.

5. Reading status

The functions *E1735A_ButtonClicked* and *E1735A_BeamStrength* return the status of buttons and beam strength respectively.

6. Working with AQB counters

The AQB input signal is used to initiate sampling. There is a 15-bit modulo counter and 11-bit roll over counter for AQB signals in the E1735A. Either the AQB signal or up/down pulses can be accepted and counted by the modulo counter. This counter overflows or underflows when the count reaches modulo value of zero. A laser sample is trigged whenever the counter overflows or underflows.
The 11-bit counter is increased or decreased by one when the modulo counter overflows or underflows. A hysteresis value between 0 and 15 can be set by the user to stabilize the overflow and underflow.
Call *E1735A_SetupAQB* to setup modulo, hysteresis, AQB or up/down mode, enable or disable AQB sampling and reset the AQB counters.
Call *E1735A_ReadAQB* to get the current count.
Call *E1735A_ReadSampleAndAQB* to get the current laser sample and count simultaneously.

7. Making a time base measurement

The timer of the E1735A can be configured to make time base measurements by the following steps:

- Arm timer and prepare for start
  Disable other triggers with *E1735A_SetupAQB* and *E1735A_StopExternalSampling*.
  Setup interval by calling *E1735A_SetupTimer*.
  Clear buffers using *E1735A_ReadAllSamples*.
- Check start condition
  To start by laser position, read current position via *E1735A_ReadSample*.
  To start by remote button clicking, get button status via *E1735A_ButtonClicked*.
  To start by software, call *E1735A_StartTimer* immediately.
- Start sampling
  Call *E1735A_StartTimer*
- Collect samples
  If the interval is greater than 0.0001s, call *E1735A_ReadSampleCount* to collect samples into the buffer and read them using *E1735A_ReadSample* or *E1735A_ReadAllSamples.* Note that *E1735A_ReadSampleCount* must be called in a timely manner to move samples from the hardware buffer to the software buffer before the hardware buffer overflows. The smaller the interval, the more frequently this function should be called.
  If the interval is smaller than 0.0001s, call *E1735A_ReadTimerSamples* to get data quickly. Note that the function won't return data until the specified sample count is reached.
- Check stop condition
  Inspect error flags of the sample, latest laser position, number of collected samples, time past or remote button status to see if the measurement should be stopped.
- Stop sampling
  Call *E1735A_StopTimer*
- Clear buffers using E1735A_ReadAllSamples.
  Please see the respective function descriptions for more details.

E1735A API DLL Summary

No.	Type	Function Name	Parameter	Page
1	int	E1735A_ReadDeviceCount	void	7
2	bool	E1735A_SelectDevice	int Index	8
3	bool	E1735A_GetAllRevisions	unsigned int* pHWRev, unsigned int* pFWRev, unsigned int* pDrvRev, unsigned int* pCoreDLLRev, unsigned int* pDLLRev	9
4	bool	E1735A_BlinkLED	void	10
5	bool	E1735A_ResetDevice	void	11
6	int	E1735A_ReadLastError	void	12
7	int	E1735A_ReadSampleCount	void	13
8	double	E1735A_ReadSample	void	14
9	int	E1735A_ReadLastTrigger	void	15
10	__int64	E1735A_ReadLastTimeStamp	void	16
11	int	E1735A_ReadAllSamples	TLaserSample* pBuf, int BufSize	17
12	bool	E1735A_SetSampleTriggers	int TriggerTypes	18
13	int	E1735A_GetSampleTriggers	void	19
14	bool	E1735A_SetupTimer	double Interval	20
15	bool	E1735A_StartTimer	void	21
16	bool	E1735A_StopTimer	void	22
17	double *	E1735A_ReadTimerSamples	int Count	23
18	bool	E1735A_SetupAQB	int Modulo, int Hysteresis, int Settings	24
19	int	E1735A_ReadAQB	void	25
20	double	E1735A_ReadSampleAndAQB	int *pAQBData	26
21	bool	E1735A_StartExternalSampling	Void	27
22	bool	E1735A_StopExternalSampling	void	28
23	int	E1735A_ReadButtonClicked	void	29
24	double	E1735A_ReadBeamStrength	void	30
25	bool	E1735A_SetOptics	int Optics	31
26	int	E1735A_GetOptics	void	32
27	bool	E1735A_SetParameter	int Index double Value	33
28	double	E1735A_GetParameter	int Index	36

The 'int' data type is four bytes long.
The 'bool' data type is 1 byte long

E1735A_ReadDeviceCount

This function returns the number of connected Agilent E1735A USB Axis Modules.

Prototype

int E1735A_ReadDeviceCount()

Parameters

Return Value

The return value is the number of E1735A(s) that are already connected when DLL is loading.

Remarks

- If a device is connected after loading of DLL, it will not be counted. If a device is unplugged after DLL is loaded, the return value will not be decreased. User should unload and reload DLL to refresh device count if any device is changed.

See Also

E1735A_SelectDevice

E1735A_SelectDevice

This function specifies an E1735A that is active and accessed by succeeding functions.

Prototype

bool E1735A_SelectDevice(int Index)

Parameters

Index: the index of E1735A, 0 for the first one, 1 for the second one, and so on.

Return Value

Value	Meaning
TRUE	Active device is changed successfully.
FALSE	Index is invalid and active device is not changed.

Remarks

- Device of index 0 is active automatically after DLL is loaded, if it is available.

E1735A_GetAllRevisions

This function returns all revision information about hardware and software.

Prototype

bool E1735A_GetAllRevisions(unsigned int* pHWRev, unsigned int* pFWRev,
unsigned int* pDrvRev, unsigned int* pCoreDLLRev, unsigned int* pDLLRev)

Parameters

Type	Name	Meaning
unsigned int*	pHWRev	Hardware revision
unsigned int*	pFWRev	Software revision
unsigned int*	pDrvRev	Revision of windows driver
unsigned int*	pCoreDLLRev	Revision of core DLL
unsigned int*	pDLLRev	Revision of this DLL

Return Value

Return if the operation is successful or not.

Remarks

- NULL pointer can be passed to this function without making any exceptions.
- Each long word revision code really consists of four 1 byte revision codes. The most significant byte is the major revision, the next byte is the minor revision, the next byte is the release number, and the least significant byte is the build number. The format is Major.Minor.Release.Build. For example a FWRev value of 0x01020304 would convert to a revision code of 1.2.3.4.

E1735A_BlinkLED

This function makes the R/W LED of active E1735A blink quickly three times.

Prototype

bool E1735A_BlinkLED()

Parameters

Return Value

Value	Meaning
TRUE	Success
FALSE	Failed to access the device.

Remarks

- This function can be used to find which E1735 is currently selected.
- The running time of this function is about 0.6 second.

See Also

E1735A_SelectDevice, E1735A_ReadLastError

E1735A_ResetDevice

This function resets hardware counters of the E1735A, clears all error flags, button click events, hardware buffer and software buffer. This function does not stop the timer if it is running. See “Remarks” before calling this function.

Prototype

bool E1735A_ResetDevice()

Parameters

Return Value

Value	Meaning
TRUE	Success
FALSE	Failed to reset the device

Remarks

- Reset will be denied if the timer is on. Call *E1735A_StopTimer* first in this case.

See Also

E1735A_SelectDevice, E1735A_ReadLastError

E1735A_ReadLastError

This function returns the error status of last operation, or '0' if no error.

Prototype

int E1735A_ReadLastError ()

Parameters

Return Value

Constant Name	Value	Meaning	Caused by Function
• **Common Error Codes**			
EC_NOERROR	0	No error found	Any
EC_UNKNOWNERROR	1	Unknown exception occurred	Any
EC_ACCESSDENIED	2	Cannot access hardware	Any
EC_BADPARAMETER	3	Parameter wrong or out of range	Any
• **Function Error Codes**			
EC_EMPTYBUFFER	11	No sample to be read	E1735A_ReadSample
EC_BUFFERFULL	12	Hardware or software buffer full	E1735A_ReadSampleCount
EC_SAMPLELOST	13	Buffer pointer is NULL	E1735A_ReadAllSamples
EC_TIMERSTILLON	14	Can't do this if timer is on.	E1735A_ResetDevice E1735A_StartTimer E1735A_ReadSampleAndAQB
EC_TIMERISOFF	15	Cannot get sample if timer is off	E1735A_ReadTimerSamples
EC_TIMEERROR	16	Duration doesn't match interval	
EC_MEMORYFULL	17	Can't allocate enough memory	
EC_NOSAMPLE	18	Software sample is disabled	E1735A_ReadSampleAndAQB
• **Laser Sample Error Codes**			
EC_LASEROFF	21	Laser head is off or warming up	E1735A_ReadSample E1735A_ReadAllSamples E1735A_ReadTimerSamples E1735A_ReadSampleAndAQB
EC_NORETURN	22	Laser beam is not returned	
EC_REFSIGLOST	23	Reference signal was broken	
EC_MEASIGLOST	24	Return beam was blocked	
EC_BADREFSIG	25	Glitch error in reference signal	
EC_BADMEASIG	26	Glitch error in return signal	
EC_IGNOREDTRIG	27	A trigger is ignored to do sampling	
EC_OUTOFRANGE	28	Laser position out of range	

Remarks

- Error EC_IGNOREDTRIG is caused when the period between two external trigs is smaller than the minimum interval (10 microseconds), or another trigger source tries to latch a sample while the timer is selecting one.
- EC_OUTOFRANGE is set when laser position moves out of ±84.93m with linear optics.

E1735A_ReadSampleCount

This function returns the number of all currently available samples, and moves all samples in the hardware buffer to the software buffer.

Prototype

int E1735A_ReadSampleCount()

Parameters

Return Value

The number of samples that is currently stored in software buffer

Remarks

- User should call this function in a timely manner to avoid overflow of the hardware buffer.
- Process samples using *E1735A_ReadAllSamples* in a timely manner to avoid buffer overflow.
- When the buffer overflows, the oldest sample will be overwritten by a new sample, whose error code is set to EC_BUFFERFULL.

See Also

E1735A_SelectDevice, E1735A_ReadLastError

E1735A_ReadSample

This function returns a laser position. If there is data in the software buffer, it is returned. Otherwise, the hardware buffer is checked and the first sample is returned, if it is present. If both buffers are empty and the software trigger is allowed, a sample is triggered and returned by software. If no sample can be returned and the software trigger is disabled, it returns a value of NAN (Not A Number) and *E1735A_ReadLastError* will return EC_EMPTYBUFFER afterwards.

Prototype

double E1735A_ReadSample()

Parameters

Return Value

The value returned depends on the interferometer type and its parameters. It returns linear displacement in millimeters as default.
The type of interferometer is set by *E1735A_SetOptics* and all related parameters can be assigned using *E1735A_SetParameter*.
NAN (not a number) may be returned if this function failed to read the sample because of an empty buffer, calculation error or hardware access denied.

Remarks

- Call *E1735A_ReadLastTimeStamp*, *E1735A_ReadLastError* or *E1735A_ReadLastTrigger* immediately after this function to get more information for this sample.
- To avoid extra software-trigged samples, call *E1735A_ReadSampleCount* to check buffer status before reading a sample.
- If a sample is saved in a buffer but its trigger is disabled (by *E1735A_SetSample Triggers*), it will be ignored. Make sure that the desired trigger types are enabled.

See Also

E1735A_ReadLastTrigger, E1735A_ReadLastTimeStamp, E1735A_SetOptics, E1735A_SetParameter, E1735A_SetSampleTriggers, E1735A_ReadLastError

E1735A_ReadLastTrigger

This function returns the trigger of the sample that was just read.

Prototype

int E1735A_ReadLastTrigger()

Parameters

Return Value

Constant Name	Value	Meaning
TT_TIMER	0	Trigged by timer of E1735A
TT_AQBNEG	1	Trigged by underflow of AQB counter
TT_REMOTE	2	Trigged by record button of remote control
TT_EXTERNAL	3	Trigged by external sampling signal
TT_SOFTWARE	4	Trigged by software
TT_AQBPOS	5	Trigged by overflow of AQB counter
TT_NONE	255	Bad sample data or exception

See Also

E1735A_SetSampleTriggers, E1735A_GetSampleTriggers

E1735A_ReadLastTimeStamp

This function returns the time stamp of the sample that was just read.

Prototype

__int64 E1735A_ReadLastTimeStamp()

Parameters

Return Value

The returned time stamp is a 64-bit integer that represents the number of 0.1-microseconds intervals that have elapsed since 1899-12-30 00:00:00. After division by $24\times60\times60\times10^7$ (number of 0.1-microseconds of a day), the result is the number of days since December 30, 1899, and the fractional part of the result is the fraction of a day. For example, the value 2.75 represents the date time January 1, 1900 6:00:00 PM.

The time stamp is calculated by computer time and a hardware time counter of the E1735A, which is a 35-bit counter that counts every 0.1 microsecond and overflows every 3435.9738368 seconds. Therefore the user should not modify computer time while the DLL is running.

Furthermore, if laser samples are left in the hardware buffer for a long time (more than 3435 seconds), overflows of time counter may be miscounted, which will lead to an incorrect time stamp. The most significant bit (Bit 63) of the return value is set if the time stamp cannot be properly retrieved in this case.

If an unhandled exception happens, the return value will be 0.

Remarks

- Call *E1735A_ReadSampleCount* in a timely manner (at least once every 3435 seconds) to collect samples with proper time stamps without overflowing the hardware time counter.
- The type of return value can be alternatively defined as “Currency”, which is a fixed-point float that occupies 8 bytes. It is stored as a scaled 64-bit integer with the four least-significant digits implicitly representing decimal places. Therefore, its meaning becomes the number of milliseconds that has elapsed since 1899-12-30 00:00:00.

E1735A_ReadAllSamples

This function checks the hardware buffer, reads out samples to the software buffer, and then copies all available samples to the user's buffer. Copied samples are then removed from software buffer.

Prototype

int E1735A_ReadAllSamples(TLaserSample* pBuf, int BufSize)

Parameters

Type	Name	Meaning
TLaserSample*	pBuf	A pointer to user's buffer
Int	BufSize	Size of user's buffer

Type definition of TLaserSample:

```
typedef struct {
        double LaserPos;        // sample value, refer to E1735A_ReadSample
        __int64 TimeStamp;      // time stamp, refer to E1735A_ReadLastTimeStamp
        int LaserTrigger:       // trigger, refer to E1735A_ReadLastTrigger
        int LaserError;         // error code, refer to E1735A_ReadLastError
} TLaserSample;
```

Return Value

The return value is the number of samples that are actually copied. If the buffer size is smaller than the number of buffered samples, remainder samples stay in the buffer.

Remarks

- If pBuf is NULL and BufSize is greater than zero, samples will be simply removed from the software buffer. *E1735A_ReadLastError* will then return EC_SAMPLELOST. The user can ignore this error if he just wants to clear the buffer and ignore all samples intentionally.

E1735A_SetSampleTriggers

This function sets the triggers that are allowed to be stored in the software buffer.

Prototype

bool E1735A_SetSampleTriggers(int TriggerTypes)

Parameters

Type	Name	Value / Meaning
int	TriggerTypes	5 least significant bits are used as bit mask for trigger, each bit controls one type of trigger, 1 means enable, 0 means disable. Following constants can be used: 0x01 (TB_TIMER): enable/disable timer trigger 0x02 (TB_SOFTWARE): enable/disable software sample 0x04 (TB_REMOTE): enable/disable remote control record sample 0x08 (TB_AQB): enable/disable AQB sample 0x10 (TB_EXTERNAL): enable/disable external trigger

Return Value

Value	Meaning
TRUE	Trigger types are set successfully.
FALSE	Failed to set trigger types. Former setting remains.

Remarks

- Note that this function does not change the hardware status, as show in Fig.2 – it just affects sample transfer between the hardware buffer and the software buffer.

See Also

E1735A_GetSampleTriggers

E1735A_GetSampleTriggers

This function returns the trigger types that are currently allowed.

Prototype

int E1735A_GetSampleTriggers()

Parameters

Return Value

Please refer to TriggerTypes parameter of *E1735A_SetSampleTriggers*.

Remarks

See Also

E1735A_SetSampleTriggers

E1735A_SetupTimer

This function sets the interval of inner sampling timer.

Prototype

bool E1735A_SetupTimer(double Interval)

Parameters

Interval: the sampling interval in second

The interval can be set from 0.00001 second (10 microseconds) to 167.77216 seconds. It is valid only if the value can be expressed as $A \times B \times 0.00001\text{sec}$, where A and B are integer numbers between 1 and 4096. For example, 0.04097 (4097=241×17) is valid, but 0.04099 is invalid. A value that has less than three significant digits is always valid.

Return Value

Value	Meaning
TRUE	Interval is successfully set.
FALSE	Specified interval is not set.

Remarks

- The function returns FALSE if the timer is already started or the interval is invalid.

See Also

E1735A_StartTimer, E1735A_StopTimer, E1735A_ReadTimerSamples

E1735A_StartTimer

This function starts the inner sampling timer.

Prototype

bool E1735A_StartTimer()

Parameters

Return Value

Value	Meaning
TRUE	Timer is started successfully.
FALSE	Timer is not started.

Remarks

- Call *E1735A_SetupTimer* function first to set interval.
- The function returns FALSE if the timer is already started.
- The device is not allowed to reset after timer has been started. Call *E1735A_StopTimer* before *E1735A_ResetDevice* if needed.

See Also

E1735A_SetupTimer, E1735A_StopTimer, E1735A_ReadTimerSamples

E1735A_StopTimer

This function stops the inner sampling timer.

Prototype

bool E1735A_StopTimer()

Parameters

Return Value

Value	Meaning
TRUE	Timer is stopped successfully.
FALSE	Failed to stop timer.

Remarks

- There may still be some samples left in hardware buffer after the timer is stopped. Check for this using *E1735A_ReadSampleCount*.

See Also

E1735A_SetupTimer, E1735A_StartTimer, E1735A_ReadTimerSamples

E1735A_ReadTimerSamples

This function reads the samples directly from the hardware buffer when the timer is on.

Prototype

double* E1735A_ReadTimerSamples(int Count)

Parameters

Type	Name	Meaning
int	Count	The number of samples to be read

Return Value

The pointer points to where the measurement results are stored. The function returns NULL if it fails to acquire required samples, call *E1735A_ReadLastError* to check it.

Remarks

- Always call *E1735A_ReadAllSamples(null,1)* to clear both the hardware and software buffers prior to setting up and starting the timer. Failing to clear the buffers prior to starting the timer and calling the *E1735A_ReadTimerSamples* function could result in obtaining incorrect data.
- The function keeps on reading samples until the all required samples are collected. Thus, if the user specifies a large number of samples and the interval is long, it may take a long time for this function to return. The running time is about Count × Interval.
- The memory is allocated and freed by this DLL. The user must not try to free the memory that the returned pointer points to.
- To access memory safely, always check if the returned pointer is valid (not NULL) before using it.

See Also

E1735A_ReadAllSamples, E1735A_SetupTimer, E1735A_StartTimer, E1735A_StopTimer, E1735A_ReadLastError

E1735A_SetupAQB

This function sets the value of the modulo counter, hysteresis value, select AQB mode and enable/disable AQB-trigged sampling.

Prototype

bool E1735A_SetupAQB(int Modulo, int Hysteresis, int Setting)

Parameters

Type	Name	Value / Meaning
int	Modulo	This is a 15-bit value that is loaded into the 15-bit modulo counter to set the count at which it overflows. Thus the counter counts from 0 to this value, which results in a modulo+1 count cycle. The preloaded value is 1023. Only the 15 least significant bits of the parameter are used and all other bits are simply ignored.
int	Hysteresis	Hysteresis is built in to the 15-bit modulo counter to avoid unexpected sampling when the input signal is not stable. The range is 0 to 15, therefore only the 4 least significant bits of the parameter are accepted and all other bits are simply ignored. The default value is 3.
int	Setting	0 (SC_DISABLEAQB): Sampling disabled, AQB mode 1 (SC_ENABLEAQB): Sampling enabled, AQB mode 2 (SC_DISABLEUPDN): Sampling disabled, Up/down pulse mode 3 (SC_ENABLEUPDN): Sampling enabled, Up/down pulse mode

Return Value

Value	Meaning
TRUE	Values are successfully set.
FALSE	Failed to write values to hardware registers.

Remarks

- When a different value is set to Modulo, the AQB counter is reset automatically.
- If AQB-trigged sampling is enabled, a sample is saved in the buffer each time the AQB counter overflows or underflows.
- The value of Hysteresis must be smaller than that of Modulo.

See Also

E1735A_ReadAQB, E1735A_ReadSampleAndAQB

E1735A_ReadAQB

This function returns the current AQB (Up/Down) count, which is calculated by both modulo counter and overflow counter.

Prototype

int E1735A_ReadAQB()

Parameters

Return Value

$$ReturnValue = ModuloCounter + (ModuloValue + 1) \times OverflowCounter$$

Remarks

- Modulo value is set by *E1735A_SetupAQB* function. When the AQB modulo counter counts to the modulo value, it overflows and returns to zero next time it is increased. When it counts down to zero, it underflows and goes to the modulo value if it is decreased. A laser sample is trigged each moment that the counter overflows or underflows, if AQB sampling is enabled (by *E1735A_SetupAQB*). Thus, samples are obtained every Modulo+1 counts.
- The AQB modulo counter counts regardless whether or not AQB sampling is enabled.
- Every time the AQB modulo counter overflows or underflows, the overflow counter is increased or decreased by one. The overflow counter is 11 bit, so it also overflows or underflows every 2048 counts.

See Also

E1735A_SetupAQB, E1735A_ReadSampleAndAQB

E1735A_ReadSampleAndAQB

This function returns the current laser position and AQB (Up/Down) count simultaneously.

Prototype

double E1735A_ReadSampleAndAQB(int *pAQBData)

Parameters

Type	Name	Meaning
Int*	pAQBData	Pointer of the variable that receives the AQB counts. Refer to *E1735A_SetupAQB*.

Return Value

Return value is the value of the laser sample that is currently latched. Refer to the return value of *E1735A_ReadSample*.

Remarks

- A software-trigged sample is required to read laser sample and AQB count at the same time. Therefore, software sample must be enabled by calling function *E1735A_SetSampleTriggers*. Otherwise NAN will be returned and EC_NOSAMPLE is returned by *E1735A_ReadLastError*.

See Also

E1735A_SetupAQB, E1735A_ReadSample, E1735A_ReadAQB

E1735A_StartExternalSampling

This function enables hardware external sampling (from pin 8 of AQB input).

Prototype

bool E1735A_StartExternalSampling()

Parameters

Return Value

Value	Meaning
TRUE	External sampling is enabled successfully.
FALSE	Error on accessing hardware.

Remarks

- The maximum frequency of the external trigger signal is 100 kHz. If this limit is exceeded, there may be only one sample that is triggered by two or more external triggers. If so, all but the first triggers are ignored. *E1735A_ReadLastError* will return EC_IGNOREDTRIG when this sample is read.
- Call *E1735A_ReadSampleCount* in a timely manner to process samples and avoid buffer-full.

See Also

E1735A_StopExternalSampling

E1735A_StopExternalSampling

This function disables external sampling of E1735A hardware.

Prototype

bool E1735A_StopExternalSampling()

Parameters

Return Value

Value	Meaning
TRUE	External sampling is disabled successfully.
FALSE	Error on accessing hardware.

Remarks

See Also

E1735A_StartExternalSampling

E1735A_ReadButtonClicked

This function returns the button click events that have happened.

Prototype

int E1735A_ReadButtonClicked()

Parameters

Return Value

Constant Name	Value	Meaning
BC_NONE	0	None of buttons is clicked
BC_RECORD	1	Record button is clicked
BC_RESET	2	Reset button is clicked
BC_BOTH	3	Both buttons are clicked

Remarks

All button click events are automatically cleared after this function is called.

See Also

E1735A_SelectDevice

E1735A_ReadBeamStrength

Returns the beam strength of measurement channel in the laser head.

Prototype

double E1735A_ReadBeamStrength()

Parameters

Return Value

A value between 0 and 1 is returned, where 0 means that the beam strength is zero and 1 means the strongest beam strength.

Remarks

See Also

E1735A_SelectDevice

E1735A_SetOptics

This function sets the interferometer type.

Prototype

bool E1735A_SetOptics(int Optics)

Parameters

Type	Name	Meaning
int	Optics	Type of interferometer

Constants for Optics

Constant Name	Value	Meaning
OT_LINEAR	0	Linear interferometer
OT_PLANEMIRROR	1	Plane mirror interferometer
OT_HIGHRESOLUTION	2	High resolution interferometer
OT_ANGULAR	3	Angular interferometer
OT_STRAIGHTNESS	4	Straightness interferometer
OT_PARALLELISM	5	Parallelism interferometer
OT_SQUARENESS	6	Squareness interferometer
OT_WAYSTRAIGHTNESS	7	Way straightness interferometer
OT_FLATNESS	8	Flatness interferometer

Return Value

Value	Meaning
TRUE	Type of optics is changed successfully.
FALSE	Wrong parameter, type of optics is not changed.

Remarks

- This function affects all functions that return sample values.

See Also

E1735A_GetOptics

E1735A_GetOptics

This function reads back the current interferometer type.

Prototype

int E1735A_GetOptics()

Parameters

Return Value

Refer to the Index parameter of *E1735A_SetOptics* function.

Remarks

See Also

E1735A_SetOptics

E1735A_SetParameter

This function sets the interferometer parameters and environment parameters.

Prototype

bool E1735A_SetParameter(int Index, double Value)

Parameters

Type	Name	Meaning
int	Index	Index of the parameter
double	Value	Value of the parameter with default unit.

Constants for Index parameter:

Constant Name	Value	Meaning
OP_WAVELENGTH	0	Vacuum wavelength of laser head
OP_AIRTEMP	1	Air temperature
OP_AIRPRES	2	Air pressure
OP_RELHUMI	3	Relative humidity
OP_AIRCOMP	4	Wavelength compensation number
OP_MATTEMP	5	Material temperature
OP_MATEXPN	6	Material expansion coefficient
OP_MATCOMP	7	Material compensation number
OP_ALLCOMP	8	Total compensation number
OP_LASERSENSE	9	Laser direction sense
OP_SCALEFACTOR	10	Linear scaling factor
OP_EQUIVALENT	11	Equivalent number for calculation
OP_UNITSCALE	12	Scale factor of unit
OP_ARMLENGTH	13	Arm length (angular)
OP_FOOTSPACE	14	Foot space (flatness)
OP_SPLITANGLE	15	Split angle (straightness)
OP_DEADPATH	16	Dead path (linear, plane mirror)

Constants and default unit for Value parameter:

Index of Parameter	Unit	Default	Min	Max
OP_WAVELENGTH	nm	632.991354	632.99	632.992
OP_AIRTEMP	ºC	20	0	40
OP_AIRPRES	kPa	101.325	70	110
OP_RELHUMI	%	50	0	100
OP_AIRCOMP		*	0.99	1
OP_MATTEMP	ºC	20	0	40
OP_MATEXPN	ppm/ºC	11.7	0	1000
OP_MATCOMP		*	0.99	1
OP_ALLCOMP		*	0.99	1
OP_LASERSENSE		1	-1	1
OP_SCALEFACTOR		1	0.01	397
OP_EQUIVALENT		*	1E-20	1E+20
OP_UNITSCALE		0.001	1E-20	1E+20
OP_ARMLENGTH	mm	32.61	10	10000
OP_FOOTSPACE	mm	100	10	1000
OP_SPLITANGLE	arc-deg	1.5916	0.001	10
OP_DEADPATH	mm	0	0	10000

* Depends on other parameters, refer to remarks.

Return Value

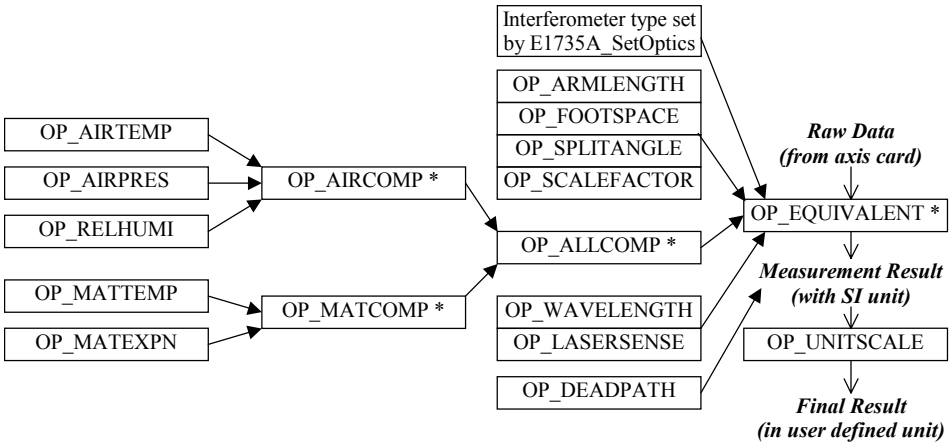
Value	Meaning
TRUE	The parameter is set successfully.
FALSE	Wrong parameter or Index, the parameter is not set.

Remarks

- All parameters of each E1735A are set as default when the DLL is loaded.
- If user tries to assign a value that is smaller than the minimum value or greater than the maximum value, function fails and the former value is unchanged.
- OP_UNITSCALE is the number that the user-selected unit is expressed in SI unit (meter, radian…). For example,

  E1735A_SetParameter(OP_UNITSCALE, 0.0254); // inch

  E1735A_SetParameter(OP_UNITSCALE, 3.14159265/180/3600); // arc-sec
- Any change made will only affect the samples read afterwards.
- The dependency of these parameters is show as below:



The calculation equation is shown as below:

$$MeasurementResult = RawData \times Equivalent - DeadPathError$$

$$FinalResult = MeasurmentResult \times UnitScale$$

It is recommended not to overwrite the value of a parameter that is labeled with *, unless the user wants to ignore the parameters that it depends on.

See Also

E1735A_GetParameter

E1735A_GetParameter

This function reads back the current value of the specified parameter.

Prototype

double E1735A_GetParameter(int Index)

Parameters

Type	Name	Meaning
int	Index	Index of the parameter Refer to Index parameter of *E1735A_SetParameter*.

Return Value

Please refer to the Value parameter of *E1735A_SetParameter* function.

Remarks

See Also

E1735A_SetParameter

Agilent Technologies

Service and Support

Contacting Agilent Technologies:

For more information about Agilent test and measurement products, applications, and services, visit our web site at www.parts.agilent.com.

Agilent's Test Measurement Fax Service for United States and Canada:

Technical information for test and measurement products and services is available 24 hours a day, 7 days a week, by calling 1-800-829-4444.

Technical Support:

If you need technical assistance with an Agilent test and measurement product or application, you can find a list of local service representatives on the web site listed above. If you do not have access to the Internet, one of the following centers can direct you to your nearest representative:

Asia Pacific:
Hong Kong SAR

Tel: (852) 2599-7777
Fax: (852) 2506-9284

Australia/New Zealand:
Blackburn, Victoria, Australia

Tel: 61 3 9210 5555

Canada:
Mississauga, ON, Canada

Tel: 877-894-4414
Fax: (905) 206-4700

Europe:
European Marketing Organization
The Netherlands

Tel: +31 20 547 2000
Fax: +31 20 547 7799

Japan:
Measurement Assistance Center
Tokyo, Japan

Tel: 81-426-56-7832
Fax: 81-426-60-8747

United States:
Test & Measurement Call Center
Englewood, CO, U.S.A.

Tel: (800) 829-4444 (Toll free in US)

Printed in U.S.A
Data subject to change
Rev 2/17/09

Continued from front matter. . .

Warranty (contd)

Agilent does not warrant that the operation of Agilent products will be uninterrupted or error free. If Agilent is unable, within a reasonable time, to repair or replace any product to a condition as warranted, customer will be entitled to a refund of the purchase price upon prompt return of the product.

Agilent products may contain remanufactured parts equivalent to new in performance or may have been subjected to incidental use.

The warranty period begins on the date of delivery or on the date of installation if installed by Agilent. If customer schedules or delays Agilent installation more than 30 days after delivery, warranty begins on the 31st day from delivery.

Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by Agilent, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.

TO THE EXTENT ALLOWED BY LOCAL LAW, THE ABOVE WARRANTIES ARE EXCLUSIVE AND NO OTHER WARRANTY OR CONDITION, WHETHER WRITTEN OR ORAL, IS EXPRESSED OR IMPLIED AND AGILENT SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Agilent will be liable for damage to tangible property per incident up to the greater of $300,000 or the actual amount paid for the product that is the subject of the claim, and for damages for bodily injury or death, to the extent that all such damages are determined by a court of competent jurisdiction to have been directly caused by a defective Agilent product.

TO THE EXTENT ALLOWED BY LOCAL LAW, THE REMEDIES IN THIS WARRANTY STATEMENT ARE CUSTOMER'S SOLE AND EXCLUSIVE REMEDIES. EXCEPT AS INDICATED ABOVE, IN NO EVENT WILL AGILENT OR ITS SUPPLIERS BE LIABLE FOR LOSS OF DATA OR FOR DIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL (INCLUDING LOST PROFIT OR DATA), OR OTHER DAMAGE, WHETHER BASED IN CONTRACT, TORT, OR OTHERWISE.

For consumer transactions in Australia and New Zealand: the warranty terms contained in this statement, except to the extent lawfully permitted, do not exclude, restrict or modify and are in addition to the mandatory statutory rights applicable to the sale of this product to you.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent products.

For any assistance, contact your nearest Agilent Sales and Service Office.

Safety Considerations (contd)

WARNING

INSTRUCTIONS FOR ADJUSTMENTS WHILE COVERS ARE REMOVED AND FOR SERVICING ARE FOR USE BY SERVICE-TRAINED PERSONNEL ONLY. TO AVOID DANGEROUS ELECTRIC SHOCK, DO NOT PERFORM SUCH ADJUSTMENTS OR SERVICING UNLESS QUALIFIED TO DO SO.

Acoustic Noise Emissions

LpA<47 dB at operator position, at normal operation, tested per EN 27779. All data are the results from type test.

Geräuschemission

LpA<47 dB am Arbeits platz, normaler Betrieb, geprüft nach EN 27779.
Die Angagen beruhen auf Ergebnissen von Typenprüfungen.

Electrostatic Discharge Immunity Testing

When the product is tested with 8kV AD, 4kV CD and 4kV ID according to IEC801-2, a system error may occur that may affect measurement data made during these disturbances. After these occurrences, the system self-recovers without user intervention.

Agilent Technologies

Manual Part Number E1735-90007

Printed in U.S.A, February 17, 2009