

VLDAQ

Echtzeit Messsignale aus AvibiaLine AVLX Geräten erhalten

Ersteller: Avibia GmbH
Datum: 03.09.2021
Version: 1.0.0.1

Inhaltsverzeichnis

Überblick.....	3
Was sind Rohdaten?.....	3
Streaming in Echtzeit und in HD.....	3
Streaming.....	3
Echtzeit.....	3
High Definition.....	4
VLDAQ-Bibliothek.....	4
Grundsätzlicher Programmablauf.....	4
Wichtige Hinweise.....	5
Beispielprogramm DataLogger.....	5
Datentypen.....	7
Rohdatenformat.....	7
VLD_Device.....	7
VLD_Error.....	7
VLD_SensorCallback.....	9
VLD_CallbackReturn.....	10
Funktionsreferenz.....	12
VLD_connectDevice.....	12
VLD_disconnectDevice.....	13
VLD_getSampleRate.....	14
VLD_getConversionFactorToVolt.....	14
VLD_getAvailableSensors.....	15
VLD_getAvailableTriggers.....	16
VLD_addSensor.....	17
VLD_removeSensor.....	18
VLD_removeAllSensors.....	19
VLD_enableTriggerData.....	20
VLD_disableTriggerData.....	21
VLD_enableAllTriggerData.....	22
VLD_disableAllTriggerData.....	22
VLD_enableCallback.....	23
VLD_disableCallback.....	24
VLD_enableCallbackForAll.....	25
VLD_disableCallbackForAll.....	26
VLD_startReadout.....	27
VLD_stopReadout.....	28
VLD_fillSensorBuffers.....	29
VLD_readSensor.....	31
VLD_releaseMemory.....	32

Überblick

Die AvibiaLine-Serie umfasst stationäre Schwingungsüberwachungen für das Condition Monitoring. Die AVLX-Typen sind zusätzlich noch in besonderem Maße für Forschung & Entwicklung sowie die Anwendung fortgeschrittener Analysetechniken wie KI geeignet.

Dazu stellen die Geräte eine Rohdaten-Schnittstelle bereit. Die Benutzung dieser Schnittstelle von externen Programmen aus und damit der Zugriff auf diese Rohdaten ist Gegenstand dieses Dokuments.

Was sind Rohdaten?

Unter Rohdaten verstehen wir hier zunächst die unbearbeiteten Messsignale aus den Schwingungssensoren. Diese Sensorsignale spiegeln den tatsächlichen Verlauf der Schwingungen im Zeitbereich wider. Sie können individuellen Signalverarbeitungsalgorithmen unterworfen werden, wie z.B. einer Kennwertbildung oder der Transformation in den Frequenzbereich (FFT).

Die Abtastung der Signale erfolgte unter der Beachtung des Nyquistkriteriums, so dass die Signale im Rahmen der Messgenauigkeit aliasingfrei sind.

Zusätzlich liefern die AVLX-Geräte auch zeitsynchron die Flanken aus den Triggereingängen und erlauben damit eine Ergänzung der Analysen um Drehfrequenzen und Phasenbeziehungen.

Streaming in Echtzeit und in HD

Die AVLX HD Serie ermöglicht zusammen mit der VLDAQ Bibliothek die leistungsfähigste Art der Datenübertragung überhaupt.

Streaming

Der Anwender muss sich bei AvibiaLine nicht mit vom Gerät beschränkten Zeitabschnitten zufrieden geben. AvibiaLine bietet **Streaming-Technologie**. Der Anwender startet und stoppt die Messung zu Zeitpunkten, die er für sinnvoll erachtet – dazwischen werden alle Samples von allen gewünschten Messkanälen **lückenlos** geliefert.

Echtzeit

Die Lieferung erfolgt in **Echtzeit**. Interne Zwischenpuffer vom Prozess bis zur Auslieferung an den Empfänger sind mit weniger als eine Sekunde dimensioniert, das heißt die Empfängerseite des Anwenders muss die bereitgestellten Daten entweder mindestens ebenso schnell abholen, wie sie anfallen oder genügend Speicher bereitstellen, um ihn mit dem gesamten anfallenden Datenumfang beschreiben zu lassen. Die anfallende Datenmenge für ein AVLX-Gerät im Endausbau (AVLX8 HD mit 8 Kanälen) beträgt:

8 Kanäle x 96 000 Werte pro Sekunde x 4 Byte pro Wert = ca. **3 Mbyte / Sekunde**. Diese Datenmenge kann mit heute üblicher Rechentechnik ohne Weiteres beherrscht werden.

High Definition

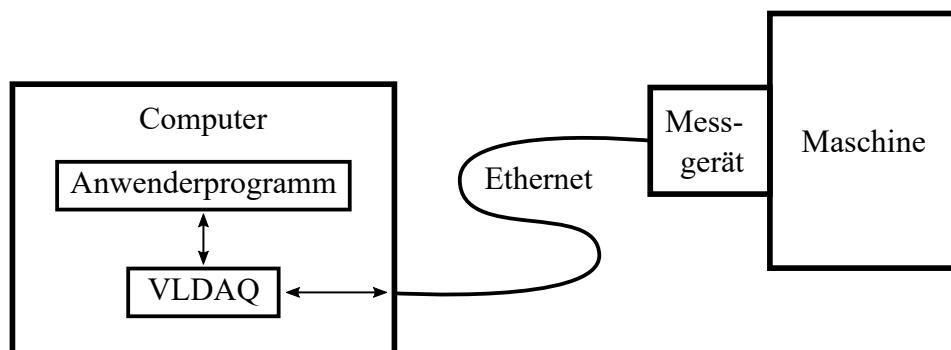
Die überdurchschnittliche Auflösung der AVLX-HD Serie bei der Abtastung

- auf der Zeitachse: 96 000 Messwerte pro Kanal und Sekunde und
- auf der Amplitudenachse: 24 Bit = 16.777.226 Stufen

bilden die Merkmale für High Definition in der Signalerfassung. Gerade im F&E Bereich oder der Erarbeitung von KI-Algorithmen halten wir den Einsatz von **HD Erfassung** für **essentiell**, weil nur sie bestimmte Effekte sichtbar macht. Diese Effekte würden mit herkömmlicher industrieller Auflösung verborgen bleiben und sich einer Auswertung entziehen.

VLDAQ-Bibliothek

Die VLDAQ-Bibliothek ermöglicht es, die Rohdaten von einem AvibiaLine-Gerät in Echtzeit auszulesen und weiterzuverarbeiten. Dazu muss das Gerät per Ethernet mit dem Computer, der die Daten empfangen soll, verbunden sein. Die Bibliothek übernimmt die Kommunikation mit dem Gerät und stellt ein einfaches Interface zur Konfiguration und Kontrolle des Auslesevorgangs bereit.



Grundsätzlicher Programmablauf

1. Bevor die Funktionen dieser Bibliothek genutzt werden können, muss zunächst eine Verbindung zu einem Gerät aufgebaut werden. Dazu dient die Funktion [VLD_connectDevice](#).
2. Sobald eine Verbindung besteht, können die verfügbaren Sensoren ([VLD_getAvailableSensors](#)) und Trigger ([VLD_getAvailableTriggers](#)) abgefragt werden. Außerdem können Sensoreigenschaften wie die Samplerate ([VLD_getSampleRate](#)) und der Umrechnungsfaktor in Volt ([VLD_getConversionFactorToVolt](#)) ermittelt werden.
3. Ein Sensor, dessen Daten ausgelesen werden sollen, wird dann mittels [VLD_addSensor](#) zum Auslesevorgang hinzugefügt. Dabei muss für jeden Sensor ein Buffer bereitgestellt werden, der die Daten zwischenspeichert.
4. Falls für einen Sensor auch Triggerdaten aufgezeichnet werden sollen, müssen diese mit [VLD_enableTriggerData](#) aktiviert werden (bzw. mit [VLD_enableAllTriggerData](#), falls alle verfügbaren Trigger genutzt werden sollen).
5. Falls die Daten mittels einer Callback-Funktion ausgelesen werden sollen, muss diese mit der Funktion [VLD_enableCallback](#) registriert werden.
6. Schritte 3-5 wiederholen, bis die gewünschte Konfiguration für den Auslesevorgang erreicht

ist. Mit den entsprechenden `remove/disable` Funktionen können Optionen auch wieder abgeschaltet werden.

7. Auslesevorgang mit [`VLD\_startReadout`](#) beginnen.
8. Falls keine Callback-Funktion hinterlegt wurde, muss [`VLD\_readSensor`](#) in einer Schleife für jeden Sensor aufgerufen werden, um zu verhindern, dass der Buffer überläuft.
9. Wenn genug Daten vorhanden sind, wird der Auslesevorgang mit [`VLD\_stopReadout`](#) beendet. Alternativ zu Schritt 7-9 kann auch [`VLD\_fillSensorBuffers`](#) aufgerufen. Diese Funktion füllt die bereitgestellten Buffer einmalig und blockiert bis alle Buffer voll sind. In diesem Fall kann der Buffer dann danach mit [`VLD\_readSensor`](#) ausgelesen werden.
10. Auslesekonfiguration nach Bedarf ändern und neue Auslesevorgänge starten, bis das Gerät nicht mehr ausgelesen werden soll. Zum Abschluss muss dann die Verbindung mit [`VLD\_disconnectDevice`](#) getrennt werden.

Wichtige Hinweise

- Die Konfiguration des AvibiaLine-Geräts erfolgt ausschließlich über den AvibiaLine Configurator. Die VLDAQ-Bibliothek kontrolliert nur, welche Sensoren und Trigger tatsächlich ausgelesen werden, nicht welche dafür zur Verfügung stehen. Insbesondere können mit [`VLD\_enableTriggerData`](#) nur Trigger aktiviert werden, die im AvibiaLine Configurator auch eingeschaltet sind. Ebenso werden Triggerschwellen sowie die Abtastrate und Kalibrierung der Sensoren über das Konfigurationsprogramm eingestellt.
- Wenn die Konfiguration des AvibiaLine-Geräts durch den AvibiaLine Configurator geändert wurde, wird auch eine bestehende Verbindung und die aktuelle Konfiguration der VLDAQ-Bibliothek für das betroffene Gerät ungültig. Die Verbindung sollte zunächst getrennt werden und muss nachdem die neue Konfiguration eingespielt wurde, neu aufgebaut werden.
- Damit die VLDAQ-Bibliothek korrekt arbeiten kann, benötigt das Anwenderprogramm Zugriff zum Netzwerk. Eine eventuell vorhandene Firewall muss das Senden und Empfangen von TCP- und UDP-Paketen auf Port 29410 erlauben.
- Es empfiehlt sich, die Verbindung zwischen Computer und AvibiaLine-Gerät so direkt wie möglich zu halten. Jeder zusätzliche Knoten (Switch, Router, etc.) stellt eine potentielle Fehlerquelle dar.

Beispielprogramm DataLogger

Mit dieser Bibliothek wird auch ein Beispielprogramm bereitgestellt. Dieses befindet sich im Ordner `ExampleDataLogger`. Dieses Programm verbindet sich mit einem Gerät und schreibt die Rohdaten in eine Datei. Das Programm wird wie folgt aufgerufen:

```
DataLogger(.exe) [IP] [Dauer s] [Präfix] [Dateinamenserweiterung]
```

Beispiel: `DataLogger.exe 192.168.0.5 10 sensor_ bin`

Im Beispiel verbindet sich das Programm zum Gerät mit der IP 192.168.0.5 und zeichnet für 10 Sekunden Daten auf. Die Daten werden für jeden Sensor in eine eigene Datei geschrieben. Der Datei-

name hat die Form `[Präfix][Sensorindex].[Dateinamenserweiterung]`. Im Beispiel also `sensor_0.bin` für den ersten Sensor.

Die Daten werden so gespeichert, wie sie von der VLDAQ-Bibliothek geliefert werden. Allerdings ist zu beachten, dass die Daten im little-endian Format, also mit den niedrigwertigsten Bytes zuerst, gespeichert werden.

Datentypen

Rohdatenformat

Bei den Rohdaten, die beispielsweise von [VLD_readSensor](#) zurückgegeben werden, handelt es sich immer um ein Array mit Einträgen vom Typ `int32_t`. Dabei entspricht jeder Eintrag genau einem Sample. AvibiaLine-Geräte benutzen intern 24 Bit breite Ganzzahlen. Diese werden um 8 Bit verschoben und stehen dann in den 24 höchstwertigsten Bits der 32 Bit Ganzzahl, die zurückgegeben wird. Die 8 niedrigwertigsten Bits werden genutzt um die Triggerdaten zu übertragen. Wenn keine Triggerdaten übertragen werden, sind die 8 niedrigwertigsten Bits alle 0. Ansonsten wird das n-te Bit gesetzt wenn für Trigger n eine Flanke registriert wird (Im Beispiel Trigger 1 und 2). Dabei wird nur auf eine steigende Flanke oder eine fallende Flanke reagiert, nicht auf beide. Die Auswahl, welche Flanke genutzt werden soll, lässt sich im AvibiaLine Configurator für jeden Trigger separat einstellen.

Beispiel:

00010101	10011110	11110110	00000011
Sensordaten			0 Trigger
			3, 2, 1

Um die Sensordaten in Volt umzurechnen wird ein Faktor benötigt, der mit der Funktion [getConversionFactorToVolt](#) ermittelt werden kann. Dieser Faktor berücksichtigt bereits die interne Verschiebung, das heißt der zurückgegebene 32-Bit Wert kann einfach mit dem Umrechnungsfaktor multipliziert werden. Falls Triggerdaten aktiviert sind, müssen allerdings die 3 niedrigwertigsten Bits manuell auf 0 gesetzt werden.

VLD_Device

`VLD_Device` repräsentiert ein spezifisches Gerät im Programm. Sobald eine Verbindung hergestellt wurde, liefert die VLDAQ-Bibliothek einen Zeiger auf ein `VLD_Device` zurück, welcher dann im weiteren Programmverlauf genutzt wird, um verschiedene Funktionen für dieses Gerät durchzuführen.

VLD_Error

`VLD_Error` ist ein Aufzählungstyp (enum), der von allen Funktionen der VLDAQ-Bibliothek genutzt wird, um einen erfolgreichen Aufruf oder aufgetretene Fehler zu signalisieren.

Nr.	Name	Beschreibung
0	<code>VLD_success</code>	Die aufgerufene Funktion war erfolgreich.

1 VLD_networkTimeout	Es gab eine Netzwerk-Zeitüberschreitung. Dieser Fehler kann auftreten, wenn es im Netzwerk zu Störungen kommt oder das Gerät nicht rechtzeitig antwortet. Auch eine falsche IP-Adresse kann zu diesem Fehler führen.
2 VLD_hostNotFound	Die angegebene IP-Adresse wurde nicht gefunden.
3 VLD_connectionRefused	Die Verbindung wurde vom Gerät abgelehnt.
4 VLD_connectionReset	Die Verbindung wurde vom Gerät zurückgesetzt. Dieser Fehler kann auftreten, wenn mehrere Verbindungen gleichzeitig zu einem Gerät aufgebaut werden.
5 VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6 VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert und auch kein neuer Auslesevorgang gestartet werden.
7 VLD_bufferOverflow	Der bereitgestellte Buffer ist voll.
8 VLD_outOfSync	Der Auslesevorgang wurde abgebrochen, da die Daten nicht mehr synchron waren. Dieser Fehler kann auftreten, wenn die Verbindung kurzzeitig unterbrochen wird oder zu viele Fehler bei der Datenübertragung auftreten.
9 VLD_receivedNoData	Ein Auslesevorgang wurde abgeschlossen, aber es wurden keine Daten empfangen. Dieser Fehler kann auch auftreten, wenn die Firewall UDP-Pakete abweist.
10 VLD_notEnoughData	Es gibt noch nicht genug neue Daten, um einen ganzen Block zu füllen.
11 VLD_unreleasedMemory	Für diesen Sensor wurden bereits Daten zur Verfügung gestellt, die noch nicht wieder freigegeben worden sind. Wenn diese Daten nicht mehr benötigt werden, müssen sie zunächst mit VLD_releaseMemory freigegeben werden, bevor neue Daten angefordert werden können.
12 VLD_noUnreleasedMemory	Es gibt für diesen Sensor keine ausstehenden Daten, die freigegeben werden könnten.
13 VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14 VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.

15 VLD_sensorAlreadyAdded	Der Sensor wurde bereits hinzugefügt.
16 VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
17 VLD_triggerInvalid	Die angegebene Triggernummer liegt außerhalb des gültigen Bereichs (0-2).
18 VLD_triggerNotAvailable	Die angegebene Triggernummer ist gültig, aber der Trigger ist geräteseitig nicht verfügbar.
19 VLD_callbackAlreadyAssigned	Für diesen Sensor wurde bereits eine Callback-Funktion hinterlegt.
20 VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.
21 VLD_bufferInvalid	Der übergebene Buffer hat eine ungültige Größe. Die Blockgröße muss mindestens 128 betragen und die Gesamtlänge des Buffers darf einen Wert von 2.000.000.000 nicht überschreiten.
22 VLD_noMemory	Der verfügbare Systemspeicher reicht nicht aus, um die Funktion auszuführen.
23 VLD_tooManyOpenFiles	Auf dem System sind nicht mehr genug Dateideskriptoren vorhanden, um die Funktion auszuführen.
24 VLD_noSystemResources	Es fehlen Systemressourcen, um die Funktion auszuführen.
25 VLD_networkBusy	Das Netzwerksystem ist überlastet.
26 VLD_networkNotAvailable	Das Netzwerksystem ist nicht verfügbar.
27 VLD_winSocketNotAvailable	Die Winsock-API ist nicht verfügbar (nur unter Microsoft Windows).

VLD_SensorCallback

Die VLDAQ-Bibliothek erlaubt es, Callback-Funktionen zu registrieren, die aufgerufen werden, sobald genügend Daten vorhanden sind. Dazu muss eine Funktion mit Signatur

```
(VLD_CallbackReturn) ( void *context,  
                        int sensorIndex,  
                        uint32_t firstSample,  
                        int32_t *data,  
                        int dataLength,  
                        bool dataContainsError)
```

definiert werden und ein Funktionszeiger an [VLD_enableCallback](#) oder [VLD_enableCallbackForAll](#) übergeben werden. VLD_SensorCallback repräsentiert diesen Funktionszeiger.

Die Callback-Funktion wird von der VLDAQ-Bibliothek mit folgenden Parametern aufgerufen:

Typ	Name	Beschreibung
void*	context	Benutzerdefinierter Kontext, der es erlaubt, der Funktion zusätzliche Daten zu übergeben. Dieser Kontext wird zusammen mit der Callback-Funktion hinterlegt und dann bei jedem Aufruf übergeben.
int	sensorIndex	Index des Sensors (0-basiert), für den die Daten zurückgegeben werden.
uint32_t	firstSample	Nummer des ersten Datensamples.
int32_t*	data	Array von Daten.
int	dataLength	Anzahl der zurückgegebenen Samples. Die Callback-Funktion wird erst aufgerufen, wenn ein kompletter Block von Daten bereitsteht. Diese Anzahl entspricht daher stets der Blockgröße, die beim Aufruf von VLD_addSensor angegeben wurde.
bool	dataContainsError	Indikator für Fehler in den zurückgegebenen Daten.

Wenn die Callback-Funktion fertig ist, muss sie ihren Status mit dem Rückgabetyt [VLD_CallbackReturn](#) an die VLDAQ-Bibliothek zurückmelden.

VLD_CallbackReturn

VLD_CallbackReturn ist ein Aufzählungstyp (enum), der als Rückgabetyt der Callback-Funktion eines Sensors genutzt wird. Mit ihm kann der VLDAQ-Bibliothek signalisiert werden, ob die Callback-Funktion erfolgreich war (VLD_callbackDone) oder ob ein Fehler aufgetreten ist (VLD_callbackFailed). In beiden Fällen werden die übergebenen Daten verworfen und ein neuer Aufruf gestartet, sobald genug neue Daten vorhanden sind. Außerdem kann signalisiert werden, dass die Callback-Funktion erfolgreich ausgeführt wurde aber die übergebenen Daten noch anderweitig benötigt werden (VLD_callbackDeferred). Dies kann beispielsweise sinnvoll sein, wenn die Callback-Funktion eine langsamere Funktion aufruft.

Nr.	Name	Beschreibung
0	VLD_callbackDone	Die Funktion wurde erfolgreich beendet und die Daten können überschrieben werden.
1	VLD_callbackFailed	Die Funktion ist gescheitert und die Daten können überschrieben werden.
2	VLD_callbackDeferred	Die Funktion wurde erfolgreich beendet aber die Daten werden noch benötigt. Für diesen Sensor wird erst wieder ein Callback aufgerufen, wenn die Daten mit VLD_releaseMemory freigegeben wurden.

Funktionsreferenz

Im Folgenden sind alle Funktionen aufgelistet, die von der VLDAQ-Bibliothek zur Verfügung gestellt werden. Die Funktionen nutzen [VLD_Error](#) als Rückgabetyt, um zu signalisieren, ob sie erfolgreich waren oder ob Fehler aufgetreten sind. Für jede Funktion sind die möglichen Rückgabewerte aufgelistet. Funktionen, die Daten zurückgeben, nutzen dazu Parameter mit Zeigertyp. In der Auflistung der Parameter ist daher auch die Spalte IO vorhanden, die angibt, ob ein Parameter als Eingabe (in), Ausgabe (out) oder beides (in, out, z.B. weil eine Eingabe modifiziert wird) fungiert.

VLD_connectDevice

Stellt eine Verbindung zu einem Gerät im Netzwerk her. Der erfolgreiche Aufruf dieser Funktion ist eine Voraussetzung für die Nutzung aller anderen Funktionen. Es wird ein [VLD_Device](#)-Objekt erzeugt, das im Folgenden zum Ansprechen des Geräts benutzt wird.

Syntax

```
VLD_Error VLD_connectDevice( const char *ip,  
                             VLD_Device **device)
```

Parameter

Typ	Name	IO	Beschreibung
const char*	ip	in	IP-Adresse des Geräts, zu dem eine Verbindung aufgebaut werden soll.
VLD_Device**	device	out	Gerät, zu dem eine Verbindung aufgebaut wurde.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
1	VLD_networkTimeout	Es gab eine Netzwerk-Zeitüberschreitung. Dieser Fehler kann auftreten, wenn es im Netzwerk zu Störungen kommt oder das Gerät nicht rechtzeitig antwortet. Auch eine falsche IP-Adresse kann zu diesem Fehler führen.
2	VLD_hostNotFound	Die angegebene IP-Adresse wurde nicht gefunden.
3	VLD_connectionRefused	Die Verbindung wurde vom Gerät abgelehnt.
4	VLD_connectionReset	Die Verbindung wurde vom Gerät zurückgesetzt. Dieser Fehler kann auftreten, wenn mehrere Verbindungen gleichzeitig zu einem Gerät aufgebaut werden.

Nr.	Name	Beschreibung
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.
22	VLD_noMemory	Der verfügbare Systemspeicher reicht nicht aus, um die Funktion auszuführen.
23	VLD_tooManyOpenFiles	Auf dem System sind nicht mehr genug Dateideskriptoren vorhanden, um die Funktion auszuführen.
24	VLD_noSystemResources	Es fehlen Systemressourcen, um die Funktion auszuführen.
25	VLD_networkBusy	Das Netzwerksystem ist überlastet.
26	VLD_networkNotAvailable	Das Netzwerksystem ist nicht verfügbar.
27	VLD_winSocketNotAvailable	Die Winsock-API ist nicht verfügbar (nur Windows).

VLD_disconnectDevice

Voraussetzung: Gerät verbunden.

Schließt die Verbindung zu einem Gerät und zerstört das entsprechende [VLD_Device](#)-Objekt. Der übergebene [VLD_Device](#)-Zeiger ist danach ungültig und muss nicht noch zusätzlich freigegeben werden.

Syntax

```
VLD_Error VLD_disconnectDevice(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, dessen Verbindung getrennt werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.
26	VLD_networkNotAvailable	Das Netzwerksystem ist nicht verfügbar.

VLD_getSampleRate

Voraussetzung: Gerät verbunden.

Ermittelt die Samplerate eines Sensors.

Syntax

```
VLD_Error VLD_getSampleRate( VLD_Device *device,  
                             int sensorIndex,  
                             int *sampleRate)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), der abgefragt werden soll.
int*	sampleRate	out	Samplerate des Sensors in Samples pro Sekunde.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_getConversionFactorToVolt

Voraussetzung: Gerät verbunden.

Ermittelt den Umrechnungsfaktor, um die übertragenen Werte in Volt umzuwandeln.

Syntax

```
VLD_Error VLD_getConversionFactorToVolt(  
        VLD_Device *device,  
        int sensorIndex,  
        float *conversionFactor)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), der abgefragt werden soll.
float*	conversionFactor	out	Umrechnungsfaktor des Sensors in Volt.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_getAvailableSensors

Voraussetzung: Gerät verbunden.

Gibt eine Liste mit auf dem Gerät verfügbaren Sensoren zurück.

Syntax

```
VLD_Error VLD_getAvailableSensors( VLD_Device *device,  
                                   const int **sensorIndexList,  
                                   int *length)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in	Gerät, das angesprochen werden soll.
const int**	sensorIndexList	out	Array das die verfügbaren Sensorindizes enthält (z.B. [0, 3], falls der 1. und 4. Sensor verfügbar ist).
int*	length	out	Länge des Arrays von verfügbaren Sensorindizes.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_getAvailableTriggers

Voraussetzung: Gerät verbunden.

Gibt eine Liste mit auf dem Gerät verfügbaren Triggern zurück.

Syntax

```
VLD_Error VLD_getAvailableTriggers(  
                                VLD_Device *device,  
                                const int **triggerIndexList,  
                                int *length)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in	Gerät, das angesprochen werden soll.
const int**	triggerIndexList	out	Array das die verfügbaren Triggerindizes enthält (z.B. [0, 1], falls der 1. und 2. Trigger verfügbar ist).
int*	length	out	Länge des Arrays von verfügbaren Triggerindizes.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_addSensor

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Fügt einen Sensor zum Auslesevorgang hinzu. Für jeden Sensor muss ein Buffer bereitgestellt werden, in den die anfallenden Daten geschrieben werden können. Bei dem Buffer handelt es sich um ein Array mit Einträgen von Typ `int32_t`. Jeder Eintrag ist ein Sample. Der Buffer ist logisch in mehrere gleich große Blöcke aufgeteilt. Daten können nur in vollständigen Blöcken gelesen werden. Die Größe des Arrays muss mindestens dem Produkt aus Anzahl der Blöcke und Blockgröße entsprechen. Wenn das Array größer ist, wird der zusätzliche Speicher nicht verwendet.

Bei der Wahl der angemessenen Blockgröße und Blockanzahl sind mehrere Punkte zu bedenken:

- Ein einzelner Block muss mindestens 128 Einträge lang sein.
- Die Gesamtlänge des Buffers (also das Produkt aus Anzahl der Blöcke und Blocklänge) darf nicht größer als 2.000.000.000 sein.
- Je kürzer ein Block ist desto häufiger müssen die Daten verarbeitet werden. Einerseits erlauben kürzere Blöcke somit eine schnellere Reaktion auf die Daten oder häufigere Aktualisierungen. Andererseits werden langsamere Operationen, die nur einmal pro Block notwendig sind (z.B. Speicherallokation oder das Öffnen und Schließen von Dateien), ebenfalls öfters aufgerufen. Generell sollten diese Funktionen aus der regelmäßigen Datenverarbeitung ausgelagert werden (z.B. indem Speicher wiederverwendet wird oder eine Datei nur einmal geöffnet wird). Falls dies nicht möglich ist, können längere Blöcke nützlich sein.
- Ein Block kann nur als Ganzes als fehlerhaft gemeldet werden. Selbst wenn nur ein kleiner Teil des Blocks tatsächlich Fehler enthält, muss der gesamte Block verworfen werden. Hier sind kürzere Blöcke von Vorteil.

Syntax

```
VLD_Error VLD_addSensor( VLD_Device *device,  
                        int sensorIndex,  
                        int32_t *buffer,  
                        int blockNumber,  
                        int blockSize)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), der hinzugefügt werden soll.
int32_t*	buffer	in	Buffer, in den die Daten geschrieben werden sollen.
int	blockNumber	in	Anzahl der Bufferblöcke.

Typ	Name	IO	Beschreibung
int	blockSize	in	Größe der Bufferblöcke.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
15	VLD_sensorAlreadyAdded	Der Sensor wurde bereits hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.
21	VLD_bufferInvalid	Der übergebene Buffer hat eine ungültige Größe. Die Blockgröße muss mindestens 128 betragen und die Gesamtlänge des Buffers darf einen Wert von 2.000.000.000 nicht überschreiten.
22	VLD_noMemory	Der verfügbare Systemspeicher reicht nicht aus, um die Funktion auszuführen.

VLD_removeSensor

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Entfernt einen Sensor aus dem Auslesevorgang.

Syntax

```
VLD_Error VLD_removeSensor( VLD_Device *device,  
                             int sensorIndex
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), der entfernt werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_removeAllSensors

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Entfernt alle hinzugefügten Sensoren aus dem Auslesevorgang.

Syntax

```
VLD_Error VLD_removeAllSensors(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_enableTriggerData

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Aktiviert die Aufzeichnung von Daten eines Triggers für einen Sensor.

Syntax

```
VLD_Error VLD_enableTriggerData( VLD_Device *device,  
                                int sensorIndex,  
                                int triggerIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), für den Triggerdaten aufgezeichnet werden sollen.
int	triggerIndex	in	Index des Triggers (0-basiert), der aufgezeichnet werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
17	VLD_triggerInvalid	Die angegebene Triggernummer liegt außerhalb des gültigen Bereichs (0-2).
18	VLD_triggerNotAvailable	Die angegebene Triggernummer ist gültig, aber der Trigger ist geräteseitig nicht verfügbar.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_disableTriggerData

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Deaktiviert die Aufzeichnung von Daten eines Triggers für einen Sensor.

Syntax

```
VLD_Error VLD_disableTriggerData( VLD_Device *device,  
                                   int sensorIndex,  
                                   int triggerIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), für den die Aufzeichnung deaktiviert werden soll.
int	triggerIndex	in	Index des Triggers (0-basiert), der nicht mehr aufgezeichnet werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
17	VLD_triggerInvalid	Die angegebene Triggernummer liegt außerhalb des gültigen Bereichs (0-2).
18	VLD_triggerNotAvailable	Die angegebene Triggernummer ist gültig, aber der Trigger ist geräteseitig nicht verfügbar.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_enableAllTriggerData

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Aktiviert die Aufzeichnung von Daten aller verfügbarer Trigger für einen Sensor.

Syntax

```
VLD_Error VLD_enableAllTriggerData( VLD_Device *device,  
                                     int sensorIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), auf dem Triggerdaten übertragen werden sollen.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_disableAllTriggerData

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Deaktiviert die Aufzeichnung von Daten aller verfügbarer Trigger für einen Sensor.

Syntax

```
VLD_Error VLD_disableAllTriggerData( VLD_Device *device,  
                                     int sensorIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), auf dem die Aufzeichnung deaktiviert werden sollen.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_enableCallback

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Legt die Callback-Funktion für einen Sensor fest. Eine bereits vorhandene Funktion wird überschrieben.

Syntax

```
VLD_Error VLD_enableCallback( VLD_Device *device,  
                              int sensorIndex,  
                              VLD_SensorCallback callback,  
                              void *context)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), für den die Callback-Funktion gesetzt werden soll.
VLD_SensorCallback	callback	in	Callback-Funktion, die genutzt werden soll.
void*	context	in	Benutzerdefinierter Kontext, der es erlaubt, der Funktion zusätzliche Daten zu übergeben.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_disableCallback

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Entfernt die Callback-Funktion von einem Sensor.

Syntax

```
VLD_Error VLD_disableCallback( VLD_Device *device,  
                               int sensorIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), dessen Callback-Funktion entfernt werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_enableCallbackForAll

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Legt die Callback-Funktion für alle mit [VLD_addSensor](#) hinzugefügten Sensoren fest. Eine bereits vorhandene Funktion wird überschrieben.

Syntax

```
VLD_Error VLD_enableCallbackForAll(  
                                VLD_Device *device,  
                                VLD_SensorCallback callback,  
                                void *context)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
VLD_SensorCallback	callback	in	Callback-Funktion, die genutzt werden soll.
void*	context	in	Benutzerdefinierter Kontext, der es erlaubt, der Funktion zusätzliche Daten zu übergeben.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_disableCallbackForAll

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Entfernt die Callback-Funktion von allen mit [VLD_addSensor](#) hinzugefügten Sensoren.

Syntax

```
VLD_Error VLD_disableCallbackForAll(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg

Nr.	Name	Beschreibung
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert werden.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_startReadout

Voraussetzung: Gerät verbunden, kein aktiver Auslesevorgang.

Startet den Auslesevorgang.

Syntax

```
VLD_Error VLD_startReadout(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
1	VLD_networkTimeout	Es gab eine Netzwerk-Zeitüberschreitung. Dieser Fehler kann auftreten, wenn es im Netzwerk zu Störungen kommt oder das Gerät nicht rechtzeitig antwortet. Auch eine falsche IP-Adresse kann zu diesem Fehler führen.
2	VLD_hostNotFound	Die angegebene IP-Adresse wurde nicht gefunden.
3	VLD_connectionRefused	Die Verbindung wurde vom Gerät abgelehnt.
4	VLD_connectionReset	Die Verbindung wurde vom Gerät zurückgesetzt. Dieser Fehler kann auftreten, wenn mehrere Verbindungen gleichzeitig zu einem Gerät aufgebaut werden.
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann kein neuer Auslesevorgang gestartet werden.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

Nr.	Name	Beschreibung
22	VLD_noMemory	Der verfügbare Systemspeicher reicht nicht aus, um die Funktion auszuführen.
23	VLD_tooManyOpenFiles	Auf dem System sind nicht mehr genug Dateideskriptoren vorhanden, um die Funktion auszuführen.
24	VLD_noSystemResources	Es fehlen Systemressourcen, um die Funktion auszuführen.
25	VLD_networkBusy	Das Netzwerksystem ist überlastet.
26	VLD_networkNotAvailable	Das Netzwerksystem ist nicht verfügbar.
27	VLD_winSocketNotAvailable	Die Winsock-API ist nicht verfügbar (nur Windows).

VLD_stopReadout

Voraussetzung: Gerät verbunden, aktiver Auslesevorgang.

Stoppt den Auslesevorgang.

Diese Funktion liefert nicht nur einen Fehler zurück, wenn der Auslesevorgang nicht gestoppt werden konnte, sondern auch, falls es während des Auslesevorgangs zu Fehlern kam.

Syntax

```
VLD_Error VLD_stopReadout(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
1	VLD_networkTimeout	Es gab eine Netzwerk-Zeitüberschreitung. Dieser Fehler kann auftreten, wenn es im Netzwerk zu Störungen kommt oder das Gerät nicht rechtzeitig antwortet. Auch eine falsche IP-Adresse kann zu diesem Fehler führen.
2	VLD_hostNotFound	Die angegebene IP-Adresse wurde nicht gefunden.

Nr.	Name	Beschreibung
4	VLD_connectionReset	Die Verbindung wurde vom Gerät zurückgesetzt. Dieser Fehler kann auftreten, wenn mehrere Verbindungen gleichzeitig zu einem Gerät aufgebaut werden.
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann die Konfiguration nicht geändert und auch kein neuer Auslesevorgang gestartet werden.
7	VLD_bufferOverflow	Der bereitgestellte Buffer ist voll.
8	VLD_outOfSync	Der Auslesevorgang wurde abgebrochen, da die Daten nicht mehr synchron waren. Dieser Fehler kann auftreten, wenn die Verbindung kurzzeitig unterbrochen wird oder zu viele Fehler bei der Datenübertragung auftreten.
9	VLD_receivedNoData	Ein Auslesevorgang wurde abgeschlossen, aber es wurden keine Daten empfangen. Dieser Fehler kann auch auftreten, wenn die Firewall UDP-Pakete abweist.

VLD_fillSensorBuffers

Voraussetzung: Gerät verbunden.

Füllt einmalig den Buffer von allen hinzugefügten Sensoren. Blockiert bis alle Buffer gefüllt sind.

Syntax

```
VLD_Error VLD_fillSensorBuffers(VLD_Device *device)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg

Nr.	Name	Beschreibung
1	VLD_networkTimeout	Es gab eine Netzwerk-Zeitüberschreitung. Dieser Fehler kann auftreten, wenn es im Netzwerk zu Störungen kommt oder das Gerät nicht rechtzeitig antwortet. Auch eine falsche IP-Adresse kann zu diesem Fehler führen.
2	VLD_hostNotFound	Die angegebene IP-Adresse wurde nicht gefunden.
3	VLD_connectionRefused	Die Verbindung wurde vom Gerät abgelehnt.
4	VLD_connectionReset	Die Verbindung wurde vom Gerät zurückgesetzt. Dieser Fehler kann auftreten, wenn mehrere Verbindungen gleichzeitig zu einem Gerät aufgebaut werden.
5	VLD_deviceNotConnected	Es besteht keine Verbindung zum Gerät.
6	VLD_readoutRunning	Es findet bereits ein Auslesevorgang statt. Bis dieser abgeschlossen ist, kann kein neuer Auslesevorgang gestartet werden.
7	VLD_bufferOverflow	Der bereitgestellte Buffer ist voll.
8	VLD_outOfSync	Der Auslesevorgang wurde abgebrochen, da die Daten nicht mehr synchron waren. Dieser Fehler kann auftreten, wenn die Verbindung kurzzeitig unterbrochen wird oder zu viele Fehler bei der Datenübertragung auftreten.
9	VLD_receivedNoData	Ein Auslesevorgang wurde abgeschlossen, aber es wurden keine Daten empfangen. Dieser Fehler kann auch auftreten, wenn die Firewall UDP-Pakete abweist.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.
22	VLD_noMemory	Der verfügbare Systemspeicher reicht nicht aus, um die Funktion auszuführen.
23	VLD_tooManyOpenFiles	Auf dem System sind nicht mehr genug Dateideskriptoren vorhanden, um die Funktion auszuführen.
24	VLD_noSystemResources	Es fehlen Systemressourcen, um die Funktion auszuführen.
25	VLD_networkBusy	Das Netzwerksystem ist überlastet.
26	VLD_networkNotAvailable	Das Netzwerksystem ist nicht verfügbar.
27	VLD_winSocketNotAvailable	Die Winsock-API ist nicht verfügbar (nur Windows).

VLD_readSensor

Voraussetzung: Gerät verbunden, der Buffer enthält genug Daten.

Liest Daten von einem Sensor. Die Anzahl der zurückgegebenen Samples entspricht der Blockgröße des Buffers. Es können nur ganze Blöcke ausgelesen werden. Falls es noch nicht genug Daten gibt wird ein Fehler ausgegeben. Sobald die Daten verarbeitet wurden, müssen sie mit [VLD_releaseMemory](#) freigegeben werden.

Syntax

```
VLD_Error VLD_readSensor( VLD_Device *device,  
                           int sensorIndex,  
                           int32_t **data,  
                           bool *dataContainsError)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), der ausgelesen werden soll.
int32_t**	data	out	Array von Daten. Die Anzahl der zurückgegebenen Samples entspricht der Blockgröße des Buffers.
bool*	dataContainsError	out	Indikator für Fehler in den zurückgegebenen Daten.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
10	VLD_notEnoughData	Es gibt noch nicht genug neue Daten, um einen ganzen Block zu füllen.
11	VLD_unreleasedMemory	Für diesen Sensor wurden bereits Daten zur Verfügung gestellt, die noch nicht wieder freigegeben worden sind. Wenn diese Daten nicht mehr benötigt werden, müssen sie zunächst mit <code>VLD_releaseMemory</code> freigegeben werden, bevor neue Daten angefordert werden können.
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt

Nr.	Name	Beschreibung
		außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
19	VLD_callbackAlreadyAssigned	Für diesen Sensor wurde bereits eine Callback-Funktion hinterlegt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.

VLD_releaseMemory

Voraussetzung: Gerät verbunden, es existieren ausstehende Daten.

Gibt ausstehende Daten eines Sensors frei.

Die VLDAQ-Bibliothek gibt dem Anwenderprogramm Zugriff auf die Messdaten im Buffer. Falls dies mittels der Funktion [VLD_readSensor](#) oder einer [Callback-Funktion](#), die `VLD_CallbackDeferred` zurückgibt, geschieht, kann die Bibliothek diesen Speicher erst dann wieder nutzen, wenn die Daten nicht mehr benötigt werden. Solange können auch keine neuen Daten für den betroffenen Sensor herausgegeben werden. `VLD_releaseMemory` erlaubt es dem Anwenderprogramm, der Bibliothek genau dies mitzuteilen.

Syntax

```
VLD_Error VLD_releaseMemory( VLD_Device *device,  
                             int sensorIndex)
```

Parameter

Typ	Name	IO	Beschreibung
VLD_Device*	device	in, out	Gerät, das angesprochen werden soll.
int	sensorIndex	in	Index des Sensors (0-basiert), für den der Speicher freigegeben werden soll.

Rückgabewerte

Typ: VLD_Error

Nr.	Name	Beschreibung
0	VLD_success	Erfolg
12	VLD_noUnreleasedMemory	Es gibt für diesen Sensor keine ausstehenden Daten, die freigegeben werden könnten.

Nr.	Name	Beschreibung
13	VLD_sensorInvalid	Die angegebene Sensornummer liegt außerhalb des gültigen Bereichs (0-7).
14	VLD_sensorNotAvailable	Die angegebene Sensornummer ist gültig, aber der Sensor ist geräteseitig nicht verfügbar.
15	VLD_sensorAlreadyAdded	Der Sensor wurde bereits hinzugefügt.
16	VLD_sensorNotAdded	Der Sensor ist im Gerät verfügbar, wurde aber noch nicht hinzugefügt.
20	VLD_pointerInvalid	Ein übergebener Zeiger ist ungültig.