

**Handbuch**

# **Vector CAN Treiber**

für LabVIEW

**Version 1.6**

**Deutsch**

## **Impressum**

Vector Informatik GmbH  
Ingersheimer Straße 24  
D-70499 Stuttgart

Die in diesen Unterlagen enthaltenen Angaben und Daten können ohne vorherige Ankündigung geändert werden. Ohne ausdrückliche schriftliche Genehmigung der Vector Informatik GmbH darf kein Teil dieser Unterlagen für irgendwelche Zwecke vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln, elektronisch oder mechanisch, dies geschieht. Alle technischen Angaben, Zeichnungen usw. unterliegen dem Gesetz zum Schutz des Urheberrechts.

© Copyright 2013, Vector Informatik GmbH  
Alle Rechte vorbehalten.

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
1.1	Zu diesem Handbuch	4
1.1.1	Zertifizierung	5
1.1.2	Gewährleistung	5
1.1.3	Warenzeichen	5
<b>2</b>	<b>CAN Treiber für LabVIEW</b>	<b>6</b>
2.1	Allgemeine Informationen	7
2.1.1	Haftung	7
2.1.2	Hard- und Softwarevoraussetzungen für den PC	7
2.1.3	Unterstützte Vector Hardware	7
2.1.4	Lieferumfang	7
2.1.5	Installation	8
2.1.6	Ausführbare LabVIEW Projekte	10
2.2	Grundlagen	11
2.2.1	Vector CAN Treiber Prinzip	12
2.2.2	Datenstrom, Blöcke und Zeitschlitz	12
2.2.3	Signalwertzuweisung	13
2.2.4	Beispiel 1 – Zuweisung mehrerer Signale	14
2.2.5	Beispiel 2 – Geringe Blockgröße und Blockfrequenz	15
2.2.6	Beispiel 3 – Geringe Blockgröße und hohe Blockfrequenz	15
2.2.7	Beispiel 4 – Blockgröße und Blockfrequenz dimensionieren	16
<b>3</b>	<b>Vector Virtual Instruments (VIs)</b>	<b>17</b>
3.1	Zugriff unter LabVIEW	18
3.1.1	Open Project	18
3.1.2	Close Project	19
3.1.3	Update Send Buffer	19
3.1.4	Trigger Send	20
3.1.5	Trigger Send Single	20
3.1.6	Get Block Size	21
3.1.7	Get Wait MS	21
3.1.8	Read	21
3.1.9	Error Code to String	22
3.1.10	Get Status	22
<b>4</b>	<b>Vector LabVIEW Configuration Tool</b>	<b>23</b>
4.1	Kurzbeschreibung	24
4.2	Baumansicht	24
4.2.1	General	24
4.2.2	CAN Channels	25
4.2.3	Signal Configuration	26
<b>5</b>	<b>Beispiel SendReceiveLoop</b>	<b>30</b>
5.1	Kurzbeschreibung	31
5.2	Starten des Beispiels	31
<b>6</b>	<b>Error Codes</b>	<b>34</b>

6.1	Übersicht	35
-----	-----------	----

# 1 Einführung

In diesem Kapitel finden Sie die folgenden Informationen:

---

1.1	Zu diesem Handbuch	Seite 4
	Zertifizierung	
	Gewährleistung	
	Warenzeichen	

---




## 1.1 Zu diesem Handbuch

### Konventionen

In den beiden folgenden Tabellen finden Sie die durchgängig im ganzen Handbuch verwendeten Konventionen in Bezug auf verwendete Schreibweisen und Symbole.

Stil	Verwendung
<b>fett</b>	Felder, Oberflächenelemente, Fenster- und Dialognamen der Software. Hervorhebung von Warnungen und Hinweisen. <b>[OK]</b> Schaltflächen in eckigen Klammern <b>File   Save</b> Notation für Menüs und Menüeinträge
<b>Windows</b>	Rechtlich geschützte Eigennamen und Randbemerkungen.
Quellcode	Dateinamen und Quellcode.
Hyperlink	Hyperlinks und Verweise.
<STRG>+<S>	Notation für Tastaturkürzel.

Symbol	Verwendung
	Dieses Symbol weist Sie auf Stellen im Handbuch hin, an denen Sie weiterführende Informationen finden.
	Dieses Symbol warnt Sie vor Gefahren, die zu Sachschäden führen können.
	Dieses Symbol weist Sie auf zusätzliche Informationen hin.
	Dieses Symbol weist Sie auf Stellen im Handbuch hin, an denen Sie Beispiele finden.
	Dieses Symbol weist Sie auf Stellen im Handbuch hin, an denen Sie Schritt-für-Schritt Anleitungen finden.
	Dieses Symbol finden Sie an Stellen, an denen Änderungsmöglichkeiten der aktuell beschriebenen Datei möglich sind.
	Dieses Symbol weist Sie auf Dateien hin, die Sie nicht ändern dürfen.

### 1.1.1 Zertifizierung

#### Qualitäts- managementsystem

Die Vector Informatik GmbH ist gemäß ISO 9001:2008 zertifiziert. Der ISO-Standard ist ein weltweit anerkannter Qualitätsstandard.

### 1.1.2 Gewährleistung

#### Einschränkung der Gewährleistung

Wir behalten uns inhaltliche Änderungen der Dokumentation und der Software ohne Ankündigung vor. Die Vector Informatik GmbH übernimmt keine Haftung für die Richtigkeit des Inhalts oder für Schäden, die sich aus dem Gebrauch der Dokumentation ergeben. Wir sind jederzeit dankbar für Hinweise auf Fehler oder für Verbesserungsvorschläge, um Ihnen in Zukunft noch leistungsfähigere Produkte anbieten zu können.

### 1.1.3 Warenzeichen

#### Geschützte Warenzeichen

Alle innerhalb der Dokumentation genannten und ggf. durch Dritte geschützten Marken- und Warenzeichen unterliegen uneingeschränkt den Bestimmungen des jeweils gültigen Kennzeichenrechts und den Besitzrechten der jeweiligen eingetragenen Eigentümer. Alle hier bezeichneten Warenzeichen, Handelsnamen oder Firmennamen sind oder können Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer sein. Alle Rechte, die hier nicht ausdrücklich gewährt werden sind vorbehalten. Aus dem Fehlen einer expliziten Kennzeichnung der in dieser Dokumentation verwendeten Warenzeichen kann nicht geschlossen werden, dass ein Name von den Rechten Dritter frei ist.

→ **Windows, Windows XP, Windows Vista, Windows 7, Windows 8** sind Warenzeichen der Microsoft Corporation.

## 2 CAN Treiber für LabVIEW

In diesem Kapitel finden Sie die folgenden Informationen:

---

2.1	Allgemeine Informationen	Seite 7
	Haftung	
	Hard- und Softwarevoraussetzungen für den PC	
	Unterstützte Vector Hardware	
	Lieferumfang	
	Installation	
	Ausführbare LabVIEW Projekte	
2.2	Grundlagen	Seite 11
	Vector CAN Treiber Prinzip	
	Datenstrom, Blöcke und Zeitschlitz	
	Signalwertzuweisung	
	Beispiel 1 – Zuweisung mehrerer Signale	
	Beispiel 2 – Geringe Blockgröße und Blockfrequenz	
	Beispiel 3 – Geringe Blockgröße und hohe Blockfrequenz	
	Beispiel 4 – Blockgröße und Blockfrequenz dimensionieren	

---



## 2.1 Allgemeine Informationen

### 2.1.1 Haftung



**Achtung:** Die Anwendung dieses Treibers kann gefährlich sein. Bitte benutzen Sie diesen Treiber mit Vorsicht.

Mit diesem Treiber sind Sie in der Lage, ein Regelsystem zu beeinflussen bzw. zu steuern. Ihre Handlungen können damit zu schwerwiegenden Personen- oder Sachschäden führen. Deshalb dürfen nur Personen diesen Treiber verwenden, welche die möglichen Konsequenzen der Aktionen mit diesem Treiber verstanden haben oder speziell für den Umgang mit diesem Treiber geschult worden sind.

**Für den Fall, dass andere Personen diesen Treiber verwenden, übernimmt die Firma Vector Informatik GmbH keine über die Fehlerbeseitigung oder die Erstattung des Kaufpreises hinausgehende Gewährleistung oder Haftung.**

### 2.1.2 Hard- und Softwarevoraussetzungen für den PC

<b>Prozessor</b>	Pentium III
<b>Arbeitsspeicher</b>	512 MB oder mehr
<b>Betriebssystem</b>	Windows XP oder Windows 7
<b>Software</b>	LabVIEW V7.0 oder höher (32 Bit)
<b>Hardware</b>	Vector Hardware mit Freischaltung für LabVIEW

### 2.1.3 Unterstützte Vector Hardware

- CAN Interfaces**
- CANcardXL
  - CANcardXLLe
  - CANboardXL / PCIe / pxi
  - CANcaseXL / log
  - VN16xx
  - VN8910 mit VN8950/VN8970

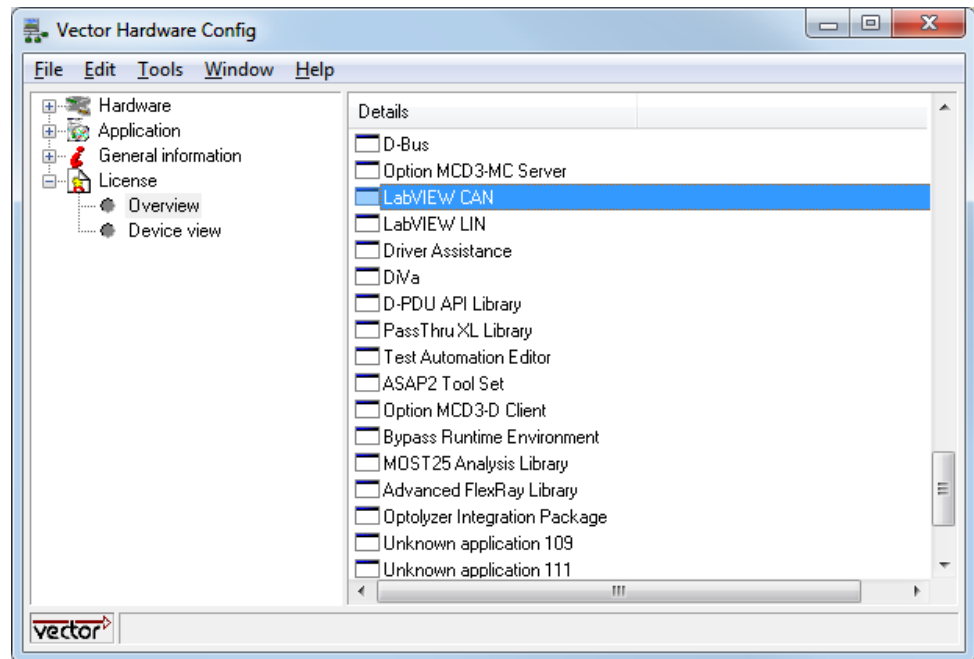
### 2.1.4 Lieferumfang

- Vector CAN Treiber `VectorLabVIEW.dll`
- Vector LabVIEW Library `VectorLabVIEW.llb`
- Beispiel `SendReceiveLoop.vi`
- Projekteditor `vlvconf.exe`
- Datenbaiseditor `CANdb++` (als Setup)
- Dieses Handbuch (PDF)

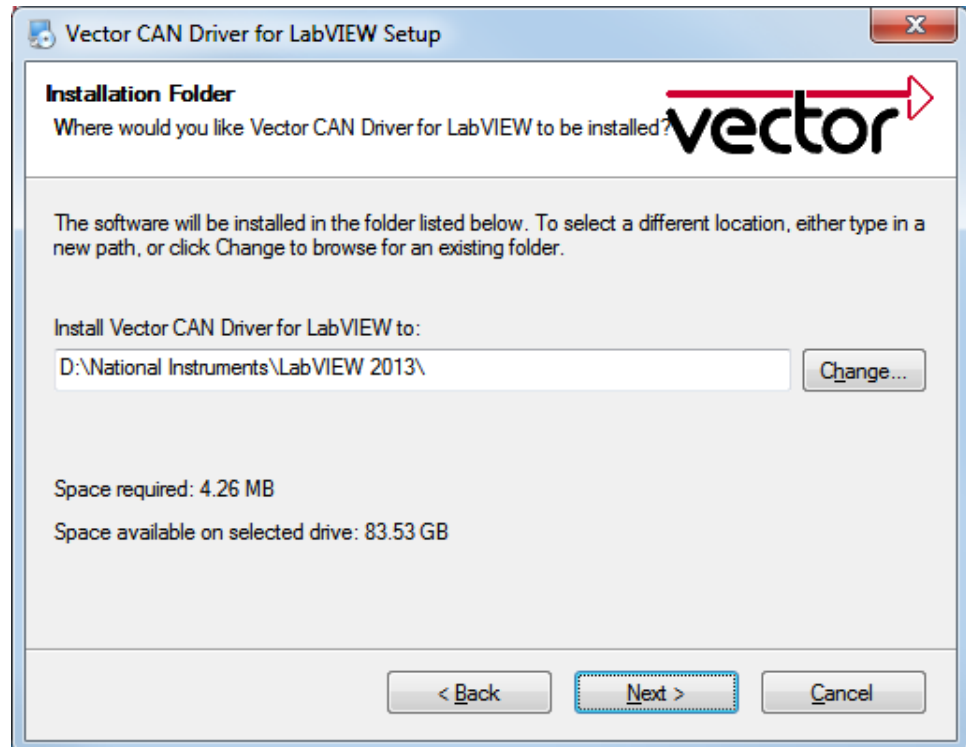
## 2.1.5 Installation



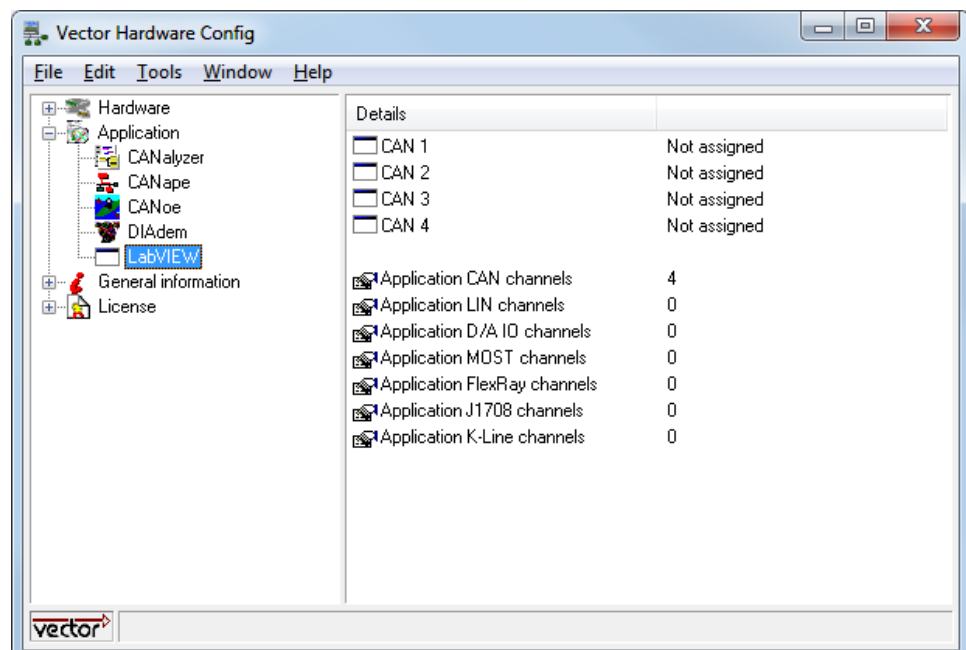
1. Vergewissern Sie sich, dass LabVIEW 7.0 oder höher (32 Bit) auf Ihrem PC installiert ist.
2. Überprüfen Sie, ob der Treiber für Ihre Vector Hardware (z.B. CANcardXL) auf Ihrem PC installiert ist. Dieser ist im Lieferumfang der Hardware enthalten.
3. Stellen Sie sicher, dass Ihre Vector Hardware über eine entsprechende LabVIEW Freischaltung verfügt. Klicken Sie auf **Start | Einstellungen | Systemsteuerung | Vector Hardware** und öffnen die Sektion **License | Overview**. Der Eintrag **LabVIEW CAN** sollte im rechten Teilfenster verfügbar sein.



4. Bitte starten Sie die Installation von der Vector Treiber-CD unter  
`\Tools\CAN Driver for LabVIEW\SetupVectorLabVIEW.exe`.
5. Folgen Sie den Anweisungen der Installations-Routine und klicken Sie auf die Schaltfläche **[Next >]**.
6. Die Installations-Routine ermittelt den Pfad der aktuellen LabVIEW-Installation selbständig. Sie können den Pfad jedoch über die Schaltfläche **[Change...]** ändern. Die Treiber-DLL sowie die Beispiele werden bei der Installation im LabVIEW Unterverzeichnis `...user.lib\express\Vector Informatik\` abgelegt.



7. Klicken Sie auf die Schaltfläche **[Next >]**, um die Installation fortzuführen.
8. Folgen Sie der Installations-Routine und beenden Sie diese mit der Schaltfläche **[Finish]**.
9. Öffnen Sie die Vector Hardware Konfiguration (**Start | Einstellungen | Systemsteuerung | Vector Hardware**) und weisen Sie der Applikation **LabVIEW** Ihre gewünschte Vector Hardware zu.



10. Vergewissern Sie sich, dass die Software-Zeitsynchronisation eingeschaltet ist. Klicken Sie auf **Start | Einstellungen | Systemsteuerung | Vector Hardware** und öffnen die Sektion **General information | Settings** und schalten **Software time synchronization** auf **YES**.

## 2.1.6 Ausführbare LabVIEW Projekte

### Vector CAN Treiber einbetten

Wenn der Einsatz des Vector CAN Treibers innerhalb eines ausführbaren LabVIEW Projekts vorgesehen ist, so sind die folgenden Dateien einzubetten:

- sLabCloseProject.vi
- sLabConvertErrCode2String.vi
- sLabGetBlockSz.vi
- sLabGetWaitMs.vi
- sLabOpenProject.vi
- sLabRead.vi
- sLabTrigSendSingle.vi
- sLabUpdateSendBuffer.vi
- CANdt2.dll
- cbdmslfho.dll
- smbsk.dll
- smbsk01.dll
- VectorLabView.dll
- xlLogAGui.dll
- xlLogAGuiSym.dll
- xlLogApi.dll

## 2.2 Grundlagen

### LabVIEW und Vector Hardware

Der Vector CAN Treiber stellt unter LabVIEW den indirekten Zugriff auf die Vector Hardware zur Verfügung und ermöglicht das Senden und Empfangen von CAN-Signalen.

Der Zugriff auf CAN-Signale erfolgt ausschließlich über symbolische Namen, die in CANdb Datenbasen definiert sind. Die Umwandlung der CAN-Rohdaten in physikalische Werte und umgekehrt erfolgt intern durch festgelegte Umrechnungsregeln in der Datenbasis.













**Hinweis:** Ändert sich die Signal-Konfiguration (z.B. Bitposition, Umrechnungsregeln), so muss lediglich die Datenbasis editiert werden.



**Hinweis:** Der Datenbasiseditor CANdb++ ist auf der Installations-CD als separates Setup zu finden.

### Funktionsumfang

Die Vector LabVIEW Library **VectorLabVIEW.lib** stellt folgende Virtual Instruments (VIs) unter LabVIEW zur Verfügung:

-  **Open Project**
-  **Close Project**
-  **Read**
-  **Update Send Buffer**
-  **Trigger Send**
-  **Trigger Send Single**
-  **Get Wait MS**
-  **Get Block Size**
-  **Convert Error Code to String**
-  **Get Status**

Eine Beschreibung der Vector VIs finden Sie im Abschnitt **Vector Virtual Instruments (VIs)** auf Seite 17.

## 2.2.1 Vector CAN Treiber Prinzip

### Datenstrom mit fester Abtaste

Der Vector CAN Treiber simuliert aus einem asynchron empfangenen CAN-Signal einen Datenstrom mit fester Abtaste. Dabei wird das empfangene Signal den Zeitstempeln entsprechend in einen Datenstrom integriert. Im Folgenden wird das Prinzip der Datenübertragung vom CAN-Bus zu LabVIEW detailliert beschrieben.

## 2.2.2 Datenstrom, Blöcke und Zeitschlitz

### Datenübertragung Treiber/LabVIEW

Ein Datenstrom ist ein kontinuierlicher Fluss von Blöcken, der durch die **Blockfrequenz** definiert ist (siehe Abschnitt General auf Seite 24). Jeder Block besteht dabei aus mindestens einem Zeitschlitz, in dem der empfangene Signalwert aufgenommen werden kann. Die Anzahl der Zeitschlitz wird durch die **Blockgröße** bestimmt (siehe Abschnitt General auf Seite 24). Je größer die gewählte Blockgröße, desto mehr Signalwerte finden in einem Block Platz.



**Hinweis:** Die Dauer eines Zeitschlitzes verkürzt sich, je größer die Blockgröße bei konstanter Blockfrequenz gewählt wird.

### Datenstrom

Die folgenden Abbildungen zeigen verschiedene Datenströme, abhängig von der gewählten Blockfrequenz und Blockgröße:

#### Ausgangsbeispiel:

Blockgröße 4  
(vier Zeitschlitz)

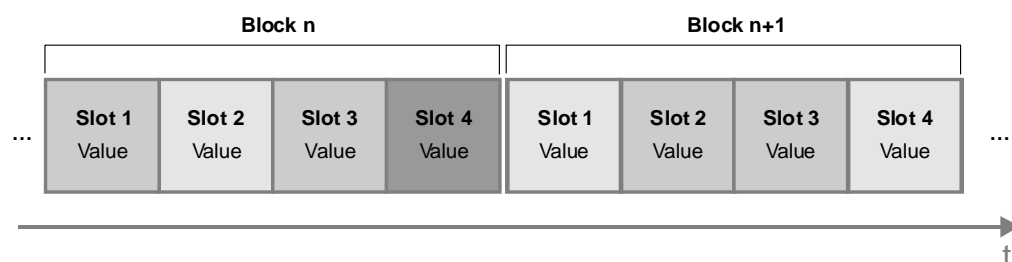


Abbildung 1: Datenstrom bei einer Blockgröße von 4.

#### Variation 1:

Doppelte  
Blockfrequenz

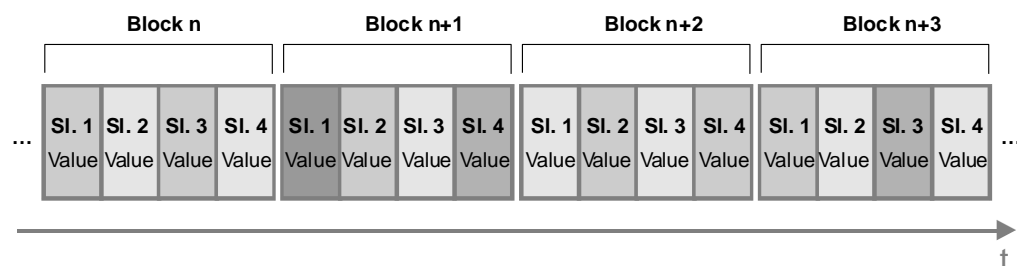


Abbildung 2: Datenstrom bei einer Blockgröße von 4, jedoch mit doppelter Blockfrequenz.

#### Variation 2:

Doppelte  
Blockgröße

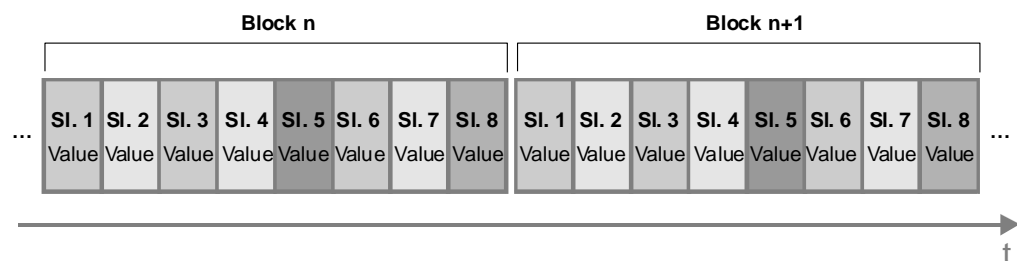


Abbildung 3: Datenstrom bei einer Blockgröße von 8. Zeitschlitz verkürzen sich.

Um die Anzahl der Datenübergaben aus dem Treiber an LabVIEW zu reduzieren, werden die einzelnen Signalwerte (einzelne Zeitschlitz) gemeinsam als Block an LabVIEW übergeben (siehe Abschnitt

Read auf Seite 21). Die Daten stehen hierbei als eindimensionales Array vom Typ Double zur Verfügung.

### 2.2.3 Signalwertzuweisung

#### Ein Datenstrom pro Signal

Ein Datenstrom verarbeitet Signalwertänderungen eines Signals, daher existiert für jedes konfigurierte Signal (siehe Abschnitt **Vector LabVIEW Configuration Tool** auf Seite 23) ein eigener Datenstrom.

#### Initialisierte Messwerte

Wenn zu Beginn einer Messung keine Signalwerte zur Verfügung stehen (CAN-Botschaft wurde nicht empfangen), wird der entsprechende Datenstrom zunächst initialisiert. D.h. jeder gelesene Block besitzt in allen Zeitschlitz einen ungültigen Wert (Null).

#### Zuweisung

Ein eintreffendes Signal auf dem CAN-Bus wird durch den Treiber in den Datenstrom integriert. Je nach aktuellem Zeitschlitz in einem Block, wird der Signalwert zugewiesen. Stehen keine aktuellen Signalwerte im zeitlichen Verlauf zur Verfügung, so wird der zuletzt gemessene Wert zum Auffüllen der übrigen Zeitschlitz genutzt.



**Beispiel:** Die folgenden Abbildungen zeigen die Arbeitsweise im Datenstrom beim Empfang eines Signals in Abhängigkeit der Blockgröße.

#### Signalwertänderungen über zwei Blöcke

Abbildung 4 zeigt eine Signalwertänderung über den CAN-Bus, bei der zwei Lesezyklen erforderlich sind.

#### Blockgröße = 2

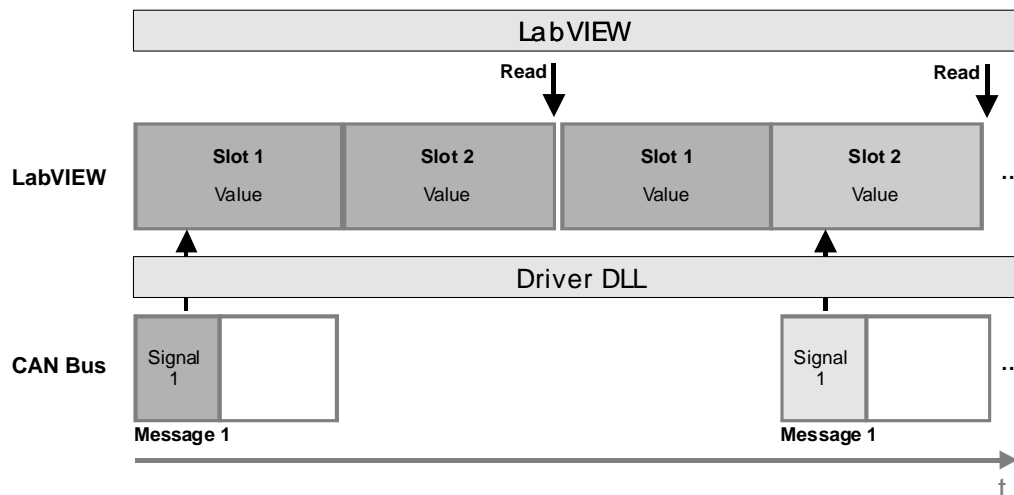


Abbildung 4: Zwei Lesezyklen.



**Hinweis:** Je kleiner die Blockgröße ist, desto länger die Zeitschlitz.

Signalwert-  
änderungen  
über einen Block

Blockgröße = 8

Abbildung 5 zeigt dieselbe Signaländerung, jedoch mit 4-facher Blockgröße und halbiertter Blockfrequenz.

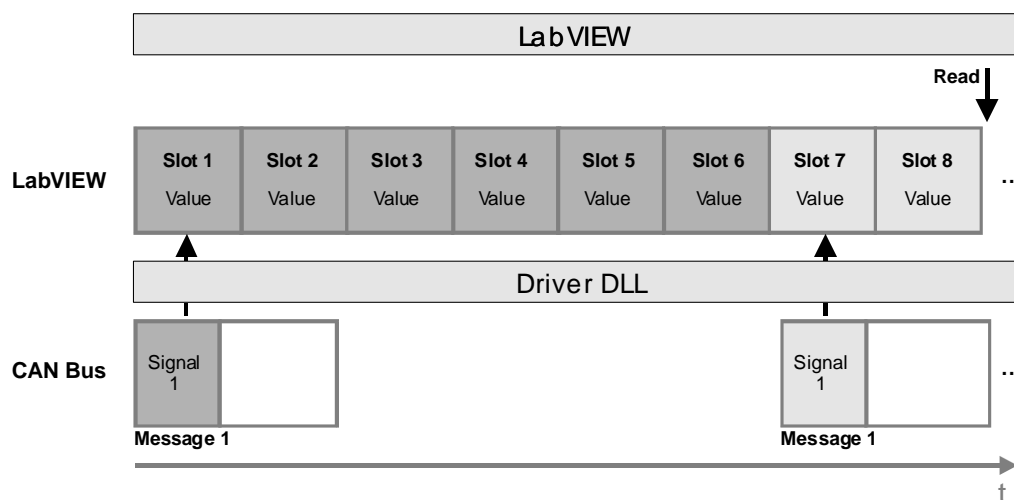


Abbildung 5: Ein Lesezyklus.

## 2.2.4 Beispiel 1 – Zuweisung mehrerer Signale

Beschreibung

Abbildung 6 zeigt die Situation bei zwei unterschiedlichen Signalen, die über den CAN-Bus gesendet werden. Pro Signal steht ein eigener Datenstrom zur Verfügung. Ungenutzte Zeitschlitze werden mit dem letzten Messwert aufgefüllt.

Blockgröße = 4

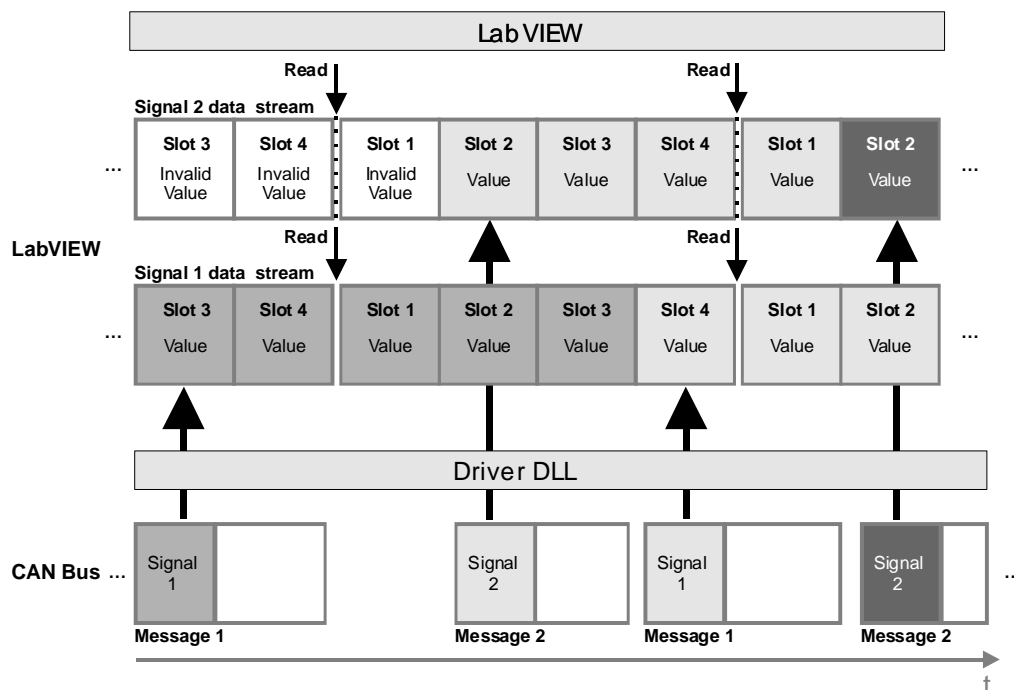


Abbildung 6: Unterschiedliche Signale aus zwei CAN-Botschaften.



## 2.2.5 Beispiel 2 – Geringe Blockgröße und Blockfrequenz

### Beschreibung

Abbildung 7 demonstriert eine zu gering eingestellte Blockgröße, wodurch die Dauer eines Schlitzes steigt. Das bedeutet, dass Signalwerte innerhalb eines Zeitschlitzes überschrieben werden können, wenn Signale zu schnell gesendet werden. Nur das zuletzt gemessene Signal innerhalb eines Zeitschlitzes ist gültig.

### Blockgröße = 2

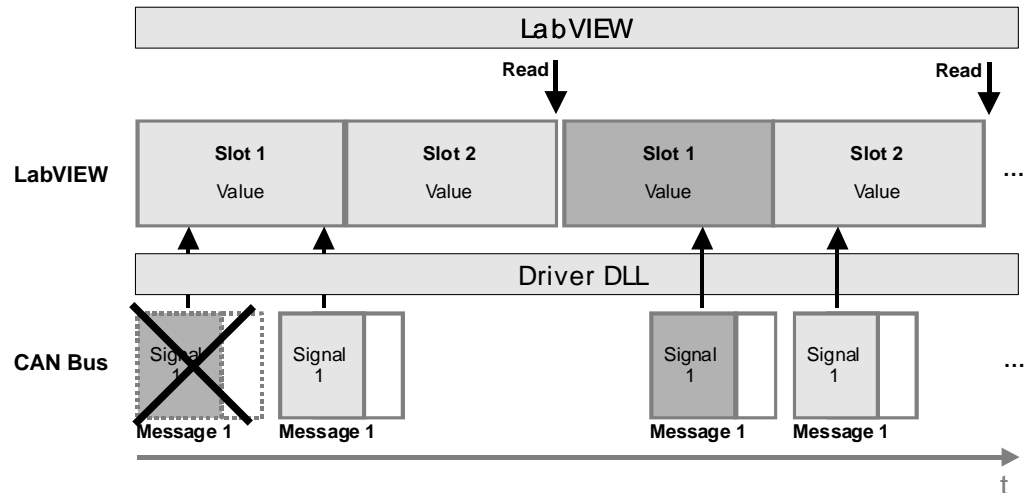


Abbildung 7: Zu geringe Blockfrequenz. Signalwertänderungen gehen verloren.

## 2.2.6 Beispiel 3 – Geringe Blockgröße und hohe Blockfrequenz

### Beschreibung

Eine schlecht dimensionierte Blockgröße und Blockfrequenz ist in Abbildung 8 dargestellt. Durch die hohe Blockfrequenz könnte eine unnötig höhere Systemlast entstehen, da die Signalwertänderungen nur durch viele Lesezyklen erfasst werden können. Im Folgenden Beispiel werden fünf zusätzliche Lesezyklen ausgeführt, obwohl keine Signaländerung vorliegt.

### Hohe Systemlast

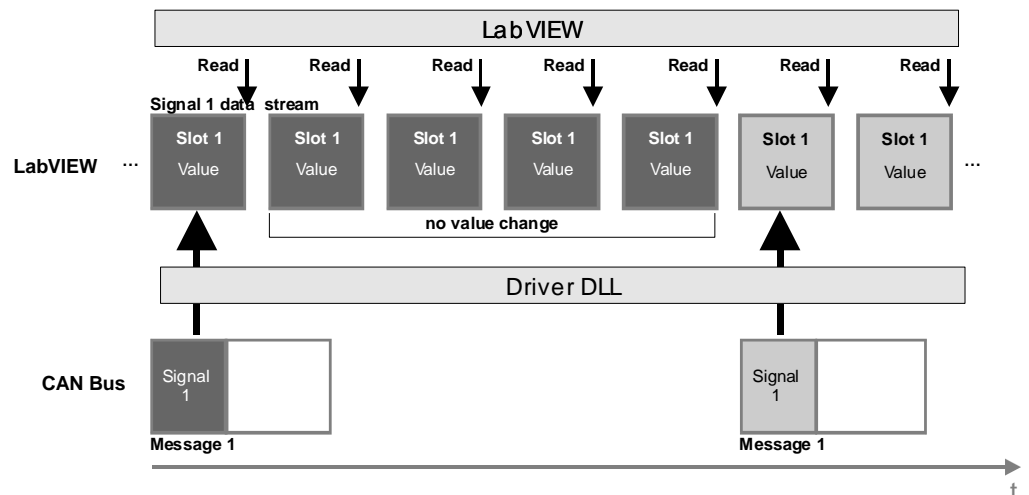


Abbildung 8: Unnötige Systemlast durch Lesezugriffe.

**Optimierung**

Um die Anzahl der Lesezyklen zu minimieren, werden die Blockgröße herauf- und die Blockfrequenz heruntergesetzt. In diesem Beispiel kann die Signalwertänderung im zweiten Lesezyklus erfasst werden. Das entspricht einem Drittel der vorherigen Systemlast durch die Read-Aufrufe.

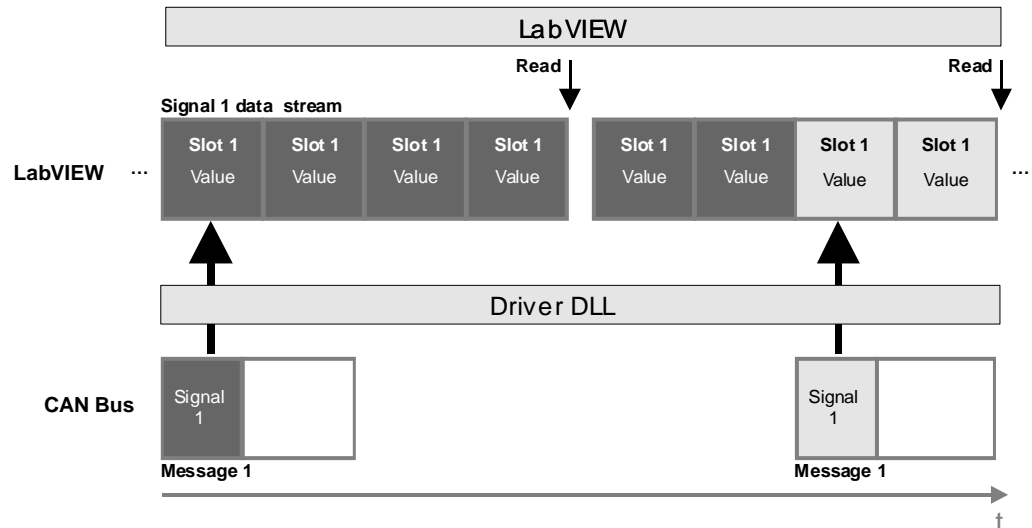
**Ausgewogene Lösung**

Abbildung 9: Alternative Lösung.

**2.2.7 Beispiel 4 – Blockgröße und Blockfrequenz dimensionieren****Tipps zur Berechnung**

- ➔ Stellen Sie fest, wie häufig Signale auf dem CAN-Bus gesendet werden. Das schnellste Signal bzw. die schnellste CAN-Botschaft sollte als Grundlage für die Dimensionierung herangezogen werden.
- ➔ Die Abtastrate (Blockfrequenz\*Blockgröße) wird 3-mal größer gewählt, als das schnellste Signal auf dem CAN-Bus.



**Beispiel:** Ein Steuergerät übermittelt die aktuelle Außentemperatur zehn Mal pro Sekunde (= 10 Hz). Um alle Signalwertänderungen zu erfassen, wird dieses 3-mal schneller abgetastet, d.h. 30 Hz.

Die Abtastfrequenz des Treibers berechnet sich aus *Blockgröße \* Blockfrequenz*.

Mögliche Einstellungen:

Blockgröße = 1, Blockfrequenz = 30 Hz oder  
Blockgröße = 3, Blockfrequenz = 10 Hz



**Beispiel:** Ein wichtiges Signal im Netzwerk wird hundert Mal pro Sekunde (= 100 Hz) gesendet. Für die Abtastung wird hier die 3-fache Frequenz gewählt (= 300 Hz).

Mögliche Einstellungen:

Blockgröße = 30, Blockfrequenz = 10 Hz oder  
Blockgröße = 15, Blockfrequenz = 20 Hz oder  
Blockgröße = 10, Blockfrequenz = 30 Hz oder  
Blockgröße = 6, Blockfrequenz = 50 Hz.

### 3 Vector Virtual Instruments (VIs)

In diesem Kapitel finden Sie die folgenden Informationen:

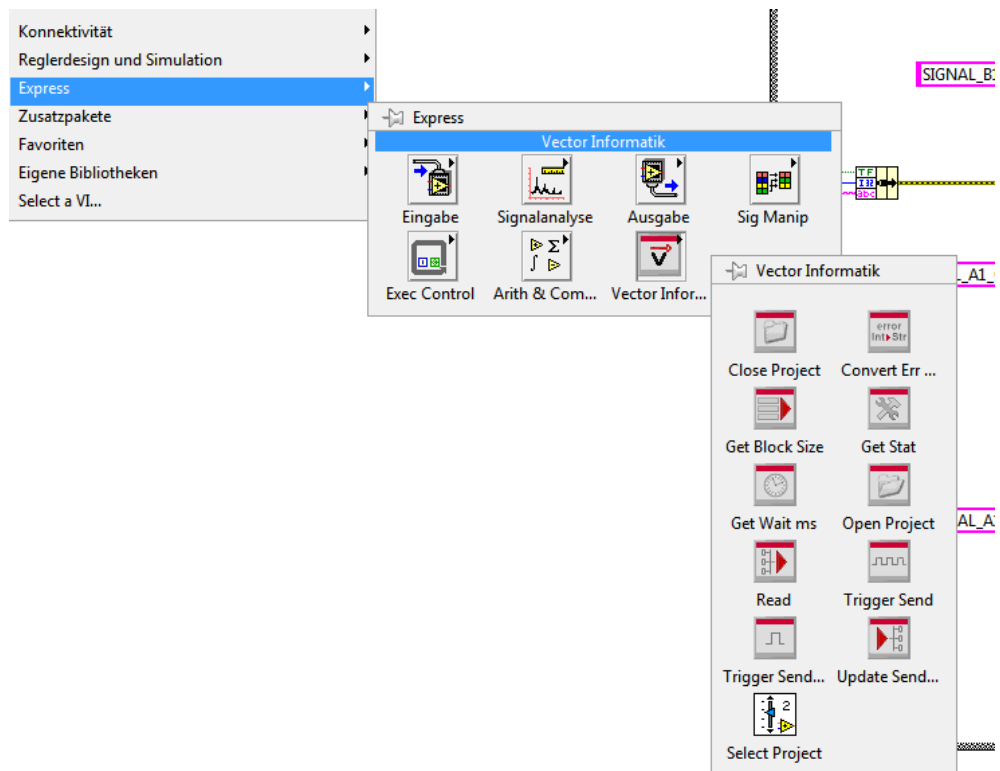
---

3.1	Zugriff unter LabVIEW	Seite 18
	Open Project	
	Close Project	
	Update Send Buffer	
	Trigger Send	
	Trigger Send Single	
	Get Block Size	
	Get Wait MS	
	Read	
	Error Code to String	
	Get Status	

---

## 3.1 Zugriff unter LabVIEW

**Eigene Bibliotheken** Nach der erfolgreichen Installation des Vector CAN Treibers, stehen Ihnen Vector VIs in der LabVIEW-Blockdiagramm-Ansicht zur Verfügung. Klicken Sie hierzu in LabVIEW in der Funktionspalette auf **Express | Vector Informatik**.



### 3.1.1 Open Project

**Symbol**



**Beschreibung**

Öffnet ein Vector LabVIEW Projekt (.vlv).

Vector LabVIEW Projekte werden mit dem mitgelieferten Projekteditor **Vector LabVIEW Config** erzeugt. Nähere Informationen hierzu finden Sie im Abschnitt **Vector LabVIEW Configuration Tool** auf Seite 23.

**Eingänge**

➔ **Project File**  
Pfad zu einem Vector LabVIEW Projekt (.vlv).

**Ausgänge**

➔ **Error**  
Ausgabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

### 3.1.2 Close Project

Symbol



Beschreibung

Schließt das geöffnete Vector LabVIEW Projekt.

Eingänge

→ **Error**  
Eingabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

Ausgänge

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

### 3.1.3 Update Send Buffer

Symbol



Beschreibung

Schreibt den Wert des selektierten CAN-Signals in den Sendepuffer.

Jedes Signal verfügt über seinen eigenen Sendepuffer; hierzu muss das Signal als **OUT** oder **OUT Single** konfiguriert sein (siehe Abschnitt **Signal Configuration** auf Seite 26).

**Update Send Buffer** kann z.B. innerhalb einer While-Schleife eingesetzt werden, um Werte zyklisch im Sendepuffer zu aktualisieren (siehe mitgeliefertes Beispiel `SendReceiveLoop.vi`).

Eingänge

- **Alias**  
Selektiertes Signal aus dem Vector LabVIEW Projekt, dessen Sendepuffer aktualisiert werden soll. Aliase lassen sich für jedes Signal im Konfigurationstool definieren (siehe Abschnitt **Vector LabVIEW Configuration Tool** auf Seite 23).
- **Error**  
Eingabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).
- **Value (double)**  
Der zu schreibende Signalwert.

Ausgänge

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

### 3.1.4 Trigger Send

#### Symbol



#### Beschreibung

Bildet aus dem Inhalt **aller Sendepuffer**, die als **OUT** (nicht Out Single!) konfiguriert sind, **mehrere** CAN Botschaften, wie sie in der Datenbasis festgelegt sind. Weitere Informationen siehe Abschnitt

Signal Configuration auf Seite 26.

**Trigger Send** kann z.B. in einer While-Schleife eingesetzt werden, um Werte aller **OUT** Sendepuffer zyklisch auf den CAN-Bus zu senden.

#### Eingänge

→ **Error**  
Eingabe Fehlercode (siehe Abschnitt Error Codes auf Seite 34).

#### Ausgänge

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt Error Codes auf Seite 34).

### 3.1.5 Trigger Send Single

#### Symbol



#### Beschreibung

Bildet aus dem Inhalt des **selektierten Sendepuffers** (siehe auch Abschnitt

Update Send Buffer auf Seite 19) **eine** entsprechende CAN Botschaft, wie sie in der Datenbasis festgelegt ist. Das Signal muss hierzu als **OUT Single** konfiguriert sein.

**Trigger Send Single** kann z.B. dafür genutzt werden, um ein einzelnes Signal gezielt auf den CAN-Bus zu senden, wenn eine bestimmte Bedingung erfüllt ist.

#### Eingänge

→ **Alias**  
Selektiertes Signal aus dem Vector LabVIEW Projekt, das gesendet werden soll. Aliase lassen sich für jedes Signal im Konfigurationstool definieren (siehe Abschnitt  
→ Signal Configuration auf Seite 26).

#### Ausgänge

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt Error Codes auf Seite 34).

### 3.1.6 Get Block Size

**Symbol****Beschreibung**

Dient lediglich als Informationsquelle und liest die im Vector LabVIEW Projekt eingestellte Blockgröße aus.

**Eingänge**

→ **Error**  
Eingabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

**Ausgänge**

→ **Blocksize**  
Numerische Ausgabe der ausgelesenen Blockgröße.

### 3.1.7 Get Wait MS

**Symbol****Beschreibung**

Ermittelt die Wartezeit aus Blockgröße und Blockfrequenz, die in While-Schleifen zum Einsatz kommen kann, z.B. als Eingang für die LabVIEW Funktion „Bis zum nächsten Vielfachen von ms warten“.

**Ausgänge**

→ **WaitMsec**  
Numerische Ausgabe in Millisekunden.

### 3.1.8 Read

**Symbol****Beschreibung**

Löst einen Zyklus aus, bei dem der aktuell ausgefüllter Block mit Werten eines Signals ausgelesen wird. Der Inhalt des Blocks (alle Zeitschlitze) steht als eindimensionales Array vom Typ Double unter LabVIEW zur Verfügung. Detaillierte Informationen zu diesem Thema finden Sie im Abschnitt **Grundlagen** auf Seite 11.

**Eingänge**

→ **Alias**  
Selektiertes Signal aus dem Vector LabVIEW Projekt. Aliase lassen sich für jedes Signal im Konfigurationstool definieren (siehe Abschnitt **Signal Configuration** auf Seite 26).

→ **Error**  
Eingabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

**Ausgänge**

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

→ **Value (double)**  
Ausgabe der empfangenen CAN-Signale (Array).

### 3.1.9 Error Code to String

#### Symbol



**Beschreibung** Konvertiert einen numerischen Fehlercode einer Vector VI in den entsprechenden Klartext.

**Eingänge** → **Error**  
Eingabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).

**Ausgänge** → **Error**  
Ausgabe Fehler in Klartext.

### 3.1.10 Get Status

#### Symbol



**Beschreibung** Dient der Bus-Kontrolle. Folgende Status können abhängig vom Eingang (Selector, SubSelector) abgefragt werden:

- **Überlauf externer Puffer (Value 0)**  
Selector = 0,  
SubSelector = x
- **Überlauf interner Puffer (Value 1)**  
Selector = 0,  
SubSelector = x
- **Anzahl fehlerfrei empf. CAN-Botschaften (Value 0)**  
Selector = 1 und  
SubSelector = 1..16 (Applikationskanal), 0 (alle Applikationskanäle)
- **Anzahl Error Frames (Value 1)**  
Selector = 1 und  
SubSelector = 1..16 (Applikationskanal), 0 (alle Applikationskanäle)

*x = don't care*

**Eingänge** → **Selector**  
Selektiert die Funktion des VIs, siehe Beschreibung oben.

→ **SubSelector**  
SubSelector abhängig von der gewählten Funktion, siehe Beschreibung oben.

**Ausgänge** → **Value 0**  
Ausgabewert abhängig von der gewählten Funktion, siehe Beschreibung oben.

→ **Value 1**  
Ausgabewert abhängig von der gewählten Funktion, siehe Beschreibung oben.

→ **Error**  
Ausgabe Fehlercode (siehe Abschnitt **Error Codes** auf Seite 34).



## 4 Vector LabVIEW Configuration Tool

In diesem Kapitel finden Sie die folgenden Informationen:

---

4.1	Kurzbeschreibung	Seite 24
4.2	Baumansicht	Seite 24
	General	
	CAN Channels	
	Signal Configuration	

---

## 4.1 Kurzbeschreibung

### Projekteditor

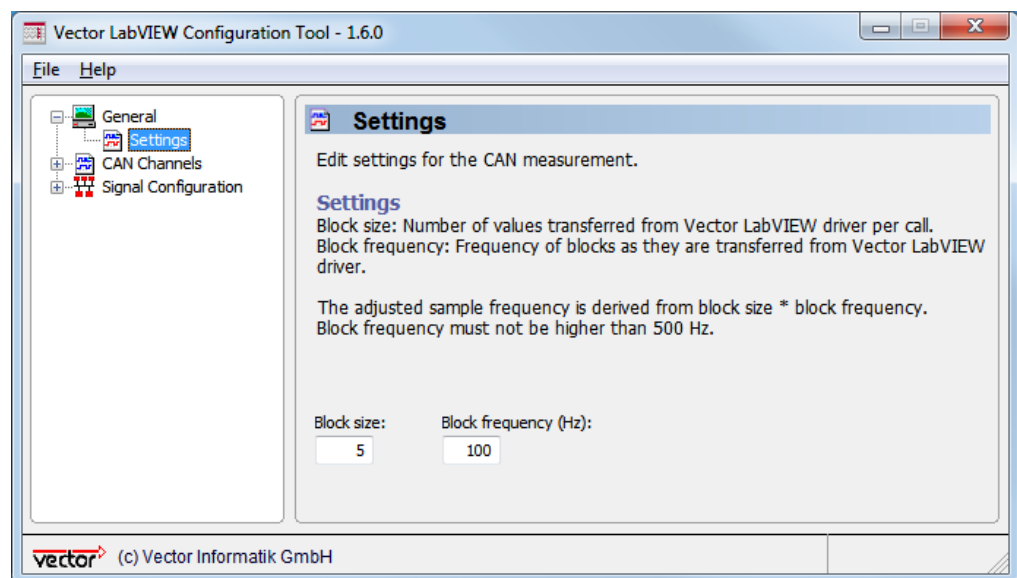
**Vector LabVIEW Configuration** ist ein Projekteditor, welcher das Erstellen von Vector LabVIEW Projekten ermöglicht. Die erstellte Projektkonfiguration wird in LabVIEW über das Vector VI **Open Project** geöffnet.

Starten Sie den Projekteditor über **Start | Programme | Vector CAN Driver for LabVIEW | Vector LabVIEW Configuration Tool**.

## 4.2 Baumansicht

### 4.2.1 General

#### Settings



#### Parameter

- **Block size**  
Anzahl der Abtastwerte (Zeitschlitz) pro Block, wobei ein Zeitschlitz genau ein Signalwert aufnehmen kann.
- **Block frequency (Hz)**  
Blockfrequenz in Hertz.



**Hinweis:** Die Abtasterate von Signalwerten ergibt sich aus:

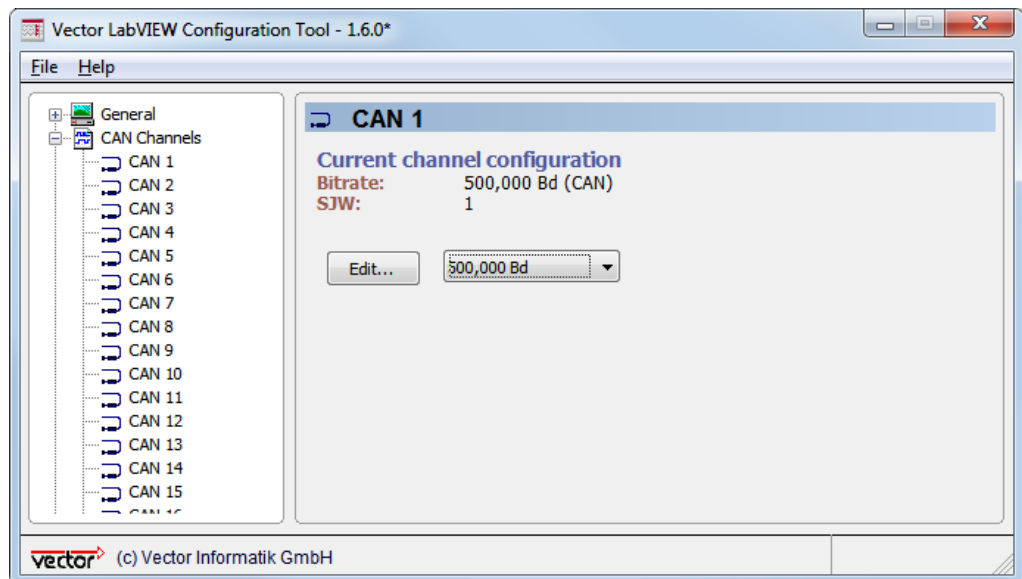
$$\text{Block size} * \text{Block frequency}$$

Detaillierte Informationen zu Blockgröße und Blockfrequenz finden Sie im Abschnitt **Grundlagen** auf Seite 11.

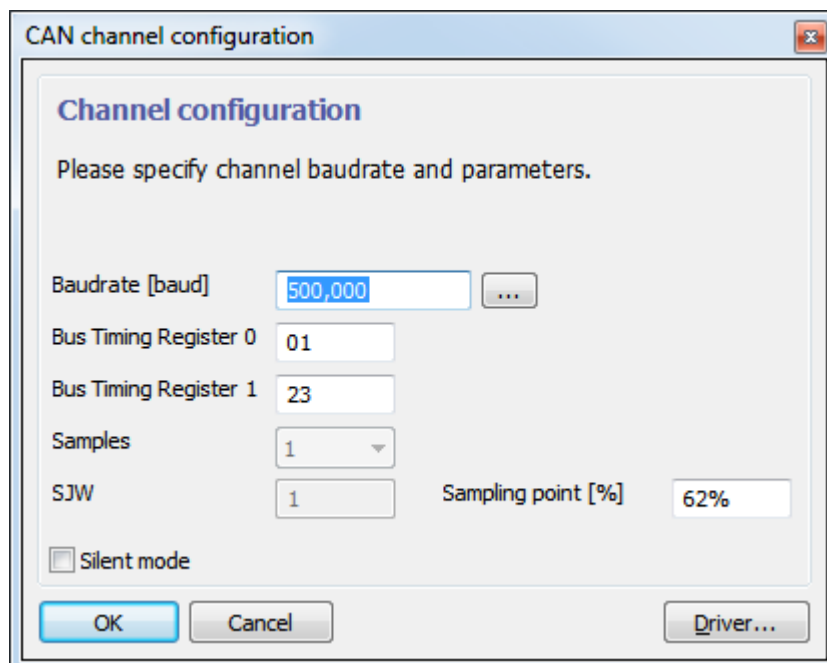
Tipps zu deren Dimensionierung erhalten Sie im Abschnitt **Beispiel 4 – Blockgröße und Blockfrequenz dimensionieren** auf Seite 16.

## 4.2.2 CAN Channels

### CAN 1...CAN 16



Der Vector LabVIEW Treiber unterstützt bis zu sechzehn CAN-Kanäle, die über die Sektion **CAN Channels** konfiguriert werden können. Hierbei stehen in einer Auswahlliste Standard-Baudraten zur Verfügung. Über die Schaltfläche **[Edit...]** gelangt man in den CAN-Konfigurationsdialog, über den eigene Baudrate angegeben werden können.

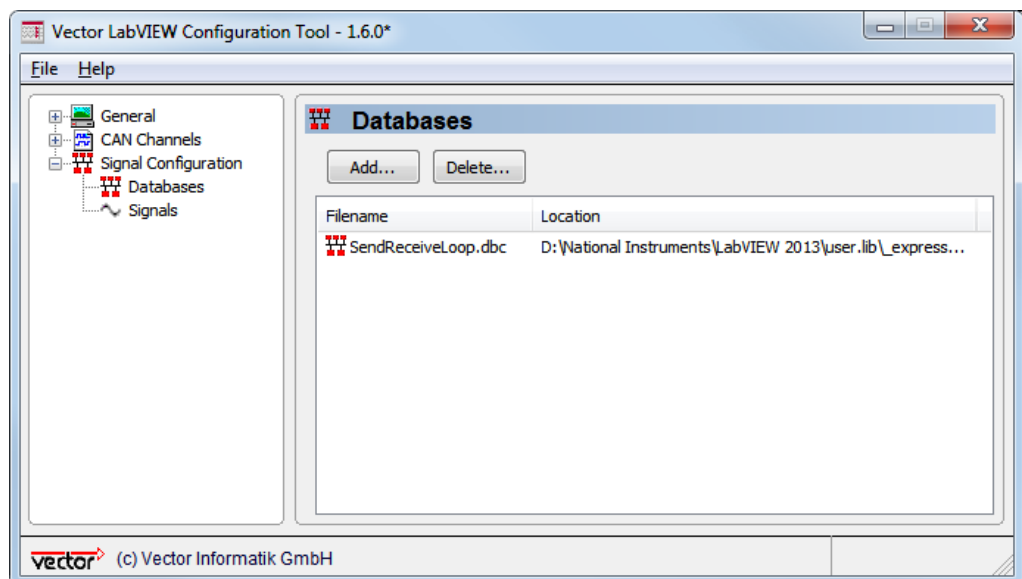


- **Baudrate [baud]**  
Baudrate des selektierten CAN Kanals.
- **Bus Timing Register 0/1**  
Experteneinstellung für Baudraten direkt über Timingregister. Beeinflusst den Sampling Point. Weitere Informationen hierzu finden Sie im Datenblatt des CAN Controllers SJA1000.

- **Sampling point [%]**  
Experteneinstellung. Verhältnis zwischen der Bitlänge zum Abtastpunkt und der Gesamtlänge der Bitzeit in Prozent. Weitere Informationen hierzu finden Sie im Datenblatt des CAN Controllers SJA1000.
- **Silent mode**  
Mit diesem Schalter lässt sich ein CAN-Kanal so konfigurieren, dass am CAN-Bus Botschaften empfangen und analysiert werden können, ohne jedoch den Bus selbst zu stören (z.B. durch ein Acknowledge).

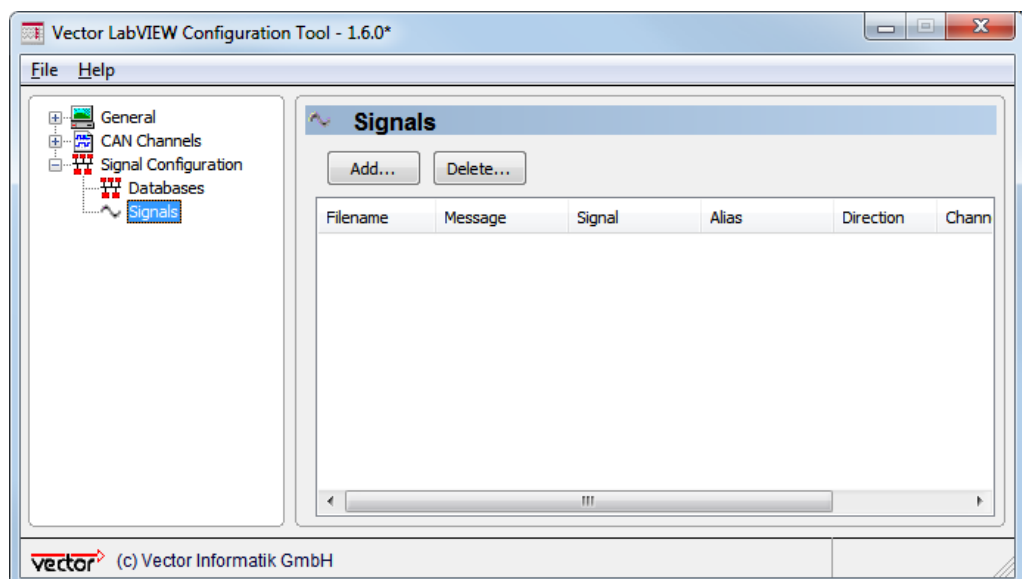
## 4.2.3 Signal Configuration

### Databases

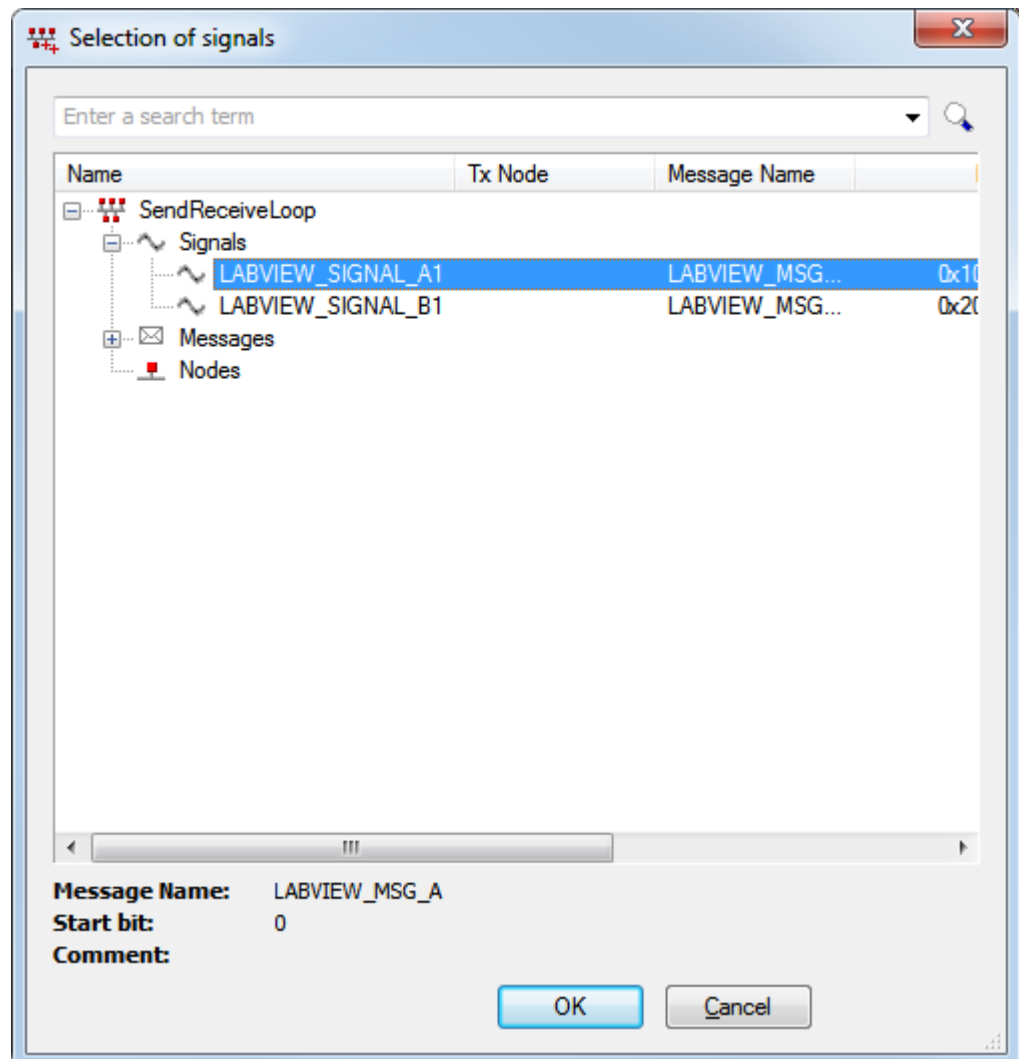


Die Signalkonfiguration ermöglicht das Hinzufügen von Signalen aus bestehenden CANdb Datenbasen (\*.dbc). In der Untersektion **Databases** müssen über die Schaltfläche **Add...** zuerst eine oder mehrere Datenbasen hinzugefügt werden. Anschließend können in der Untersektion **Signals** eine oder mehrere Signale selektiert und hinzugefügt werden, die in LabVIEW zur Verfügung stehen sollen.

### Signale



Ein Klick auf die Schaltfläche **[Add...]** öffnet den Signal-Auswahl-Dialog.



Der oberste Knoten der Baumansicht stellt die Datenbasis dar. Darunter findet man die definierten Signale und Botschaften.

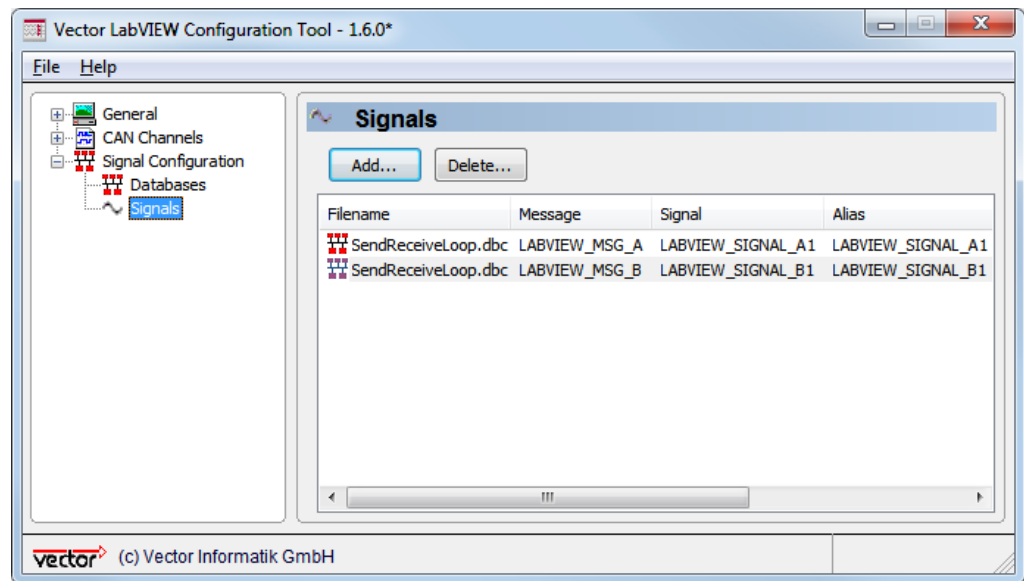
Signale und Botschaften können in einer komplexeren Datenbasis mit Hilfe der Suchfunktion gefunden werden. Über das Eingabefeld (Lupen-Symbol) lassen sich Suchstrings eingeben.



**Hinweis:** Das Sternsymbol '\*' wird als Suchstring mitberücksichtigt. Für die Suche ähnlicher Signalnamen genügt lediglich ein Textfragment, z.B. „labview\_“

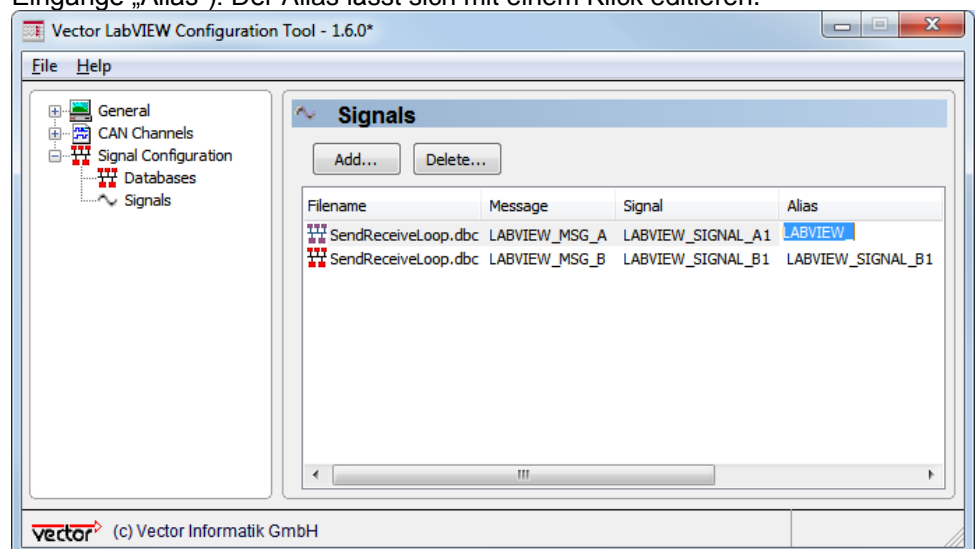
Signale selektieren  
Hinzugefügte Signale

Die Signalauswahl wird mit der Schaltfläche **[OK]** bestätigt.



Die Signalliste enthält folgende Spalten:

- ➔ **Filename**  
Name der Datenbasis.
- ➔ **Message**  
Zeigt die Botschaft an, in der sich das Signal befindet.
- ➔ **Signal**  
Zeigt den Signalnamen an.
- ➔ **Alias**  
Dient als Referenz für Vector VIs unter LabVIEW (siehe jeweils Vector VI Eingänge „Alias“). Der Alias lässt sich mit einem Klick editieren.



→ **Direction**

Definiert ein Signal als Eingang oder Ausgang. Die Richtung lässt sich mit einem Doppelklick in der Liste ändern.

**In**

Das Signal wird unter LabVIEW empfangen (Vector VI **Read**).

**Out**

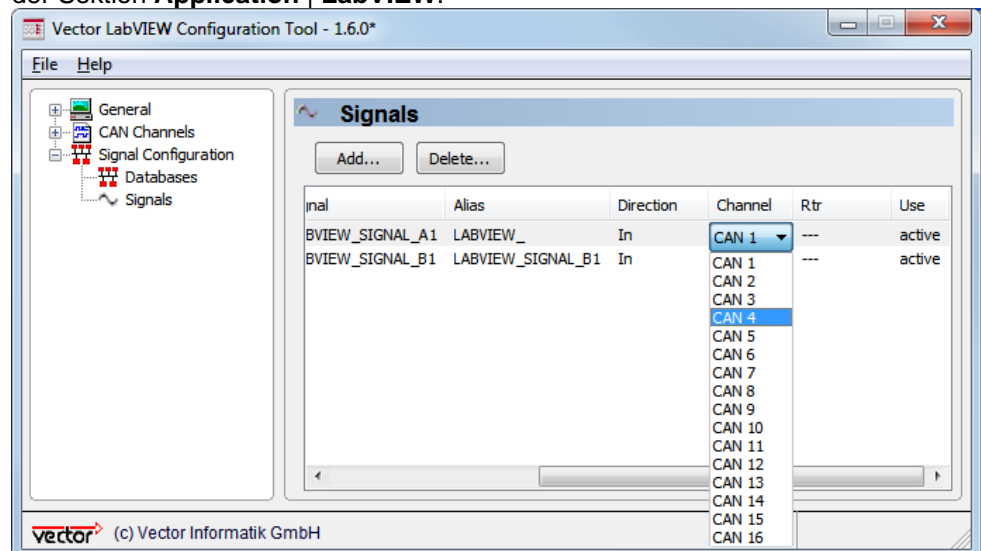
Dieses Signal, sowie alle anderen mit der Direction „Out“ werden gleichzeitig gesendet, wenn das Vector VI **SendTrigger** aufgerufen wird.

**Out Single**

Nur das selektierte Signal wird gesendet (Vector VI **SendTriggerSingle**).

→ **Channel**

CAN-Kanal, auf dem empfangen oder gesendet werden soll. Bezieht sich auf einen der (maximal sechzehn) Kanäle in der Vector Hardware Konfiguration unter der Sektion **Application | LabVIEW**.



→ **Rtr**

Remote Transmission. Der RTR Schalter wirkt sich auf alle Signale aus, die sich in derselben Botschaft befinden. D.h. ein gesetztes RTR generiert immer ein Remote Frame auf dem CAN-Bus, gleichgültig in welchem Signal (innerhalb derselben Botschaft) das RTR gesetzt ist.

→ **Use**

Um Datenströme aus großen Datenbasen unter LabVIEW auf ein Minimum zu reduzieren und damit Systemressourcen einzusparen, können mit diesem Schalter Signale ein- und ausgeschaltet werden. Für die Verwaltung von Signalwerten existiert pro Signal ein Datenstrom (siehe Abschnitt **Grundlagen** auf Seite 11). Inaktive Signale stehen unter LabVIEW **nicht** zur Verfügung und rufen eine Fehlermeldung hervor, beim Versuch diese dennoch anzusprechen.

## 5 Beispiel SendReceiveLoop

In diesem Kapitel finden Sie die folgenden Informationen:

---

5.1	Kurzbeschreibung	Seite 31
5.2	Starten des Beispiels	Seite 31

---



## 5.1 Kurzbeschreibung

### Umfang

Nach der erfolgreichen Installation des Vector CAN Treibers finden Sie im Unterverzeichnis ...\\\_express\\Vector Informatik\\examples\\ eine Beispielanwendung, bestehend aus:

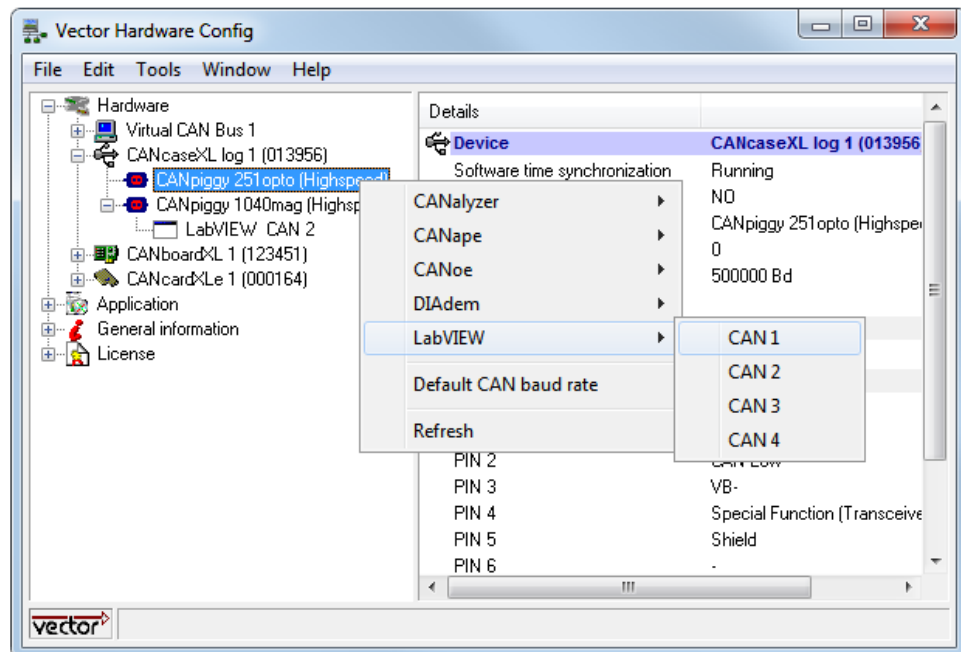
- Datenbasis SendReceiveLoop.dbc mit definierten Signalen
- Vector LabVIEW Projekt SendReceiveLoop.vlv
- LabVIEW VI SendReceiveLoop.vi

Das Beispiel nutzt alle Vector VIs und zeigt eine Loop-Anwendung, bei der simulierte Signale aus LabVIEW heraus über den ersten CAN-Kanal auf den CAN-Bus gesendet und über den zweiten CAN-Kanal empfangen werden. Gezeigt wird sowohl das zyklische als auch das Ereignis-getriggerte Senden über Vector VIs. Die Datenbasis stellt hierfür jeweils eine Botschaft mit einem Nutzsignal zur Verfügung.

## 5.2 Starten des Beispiels

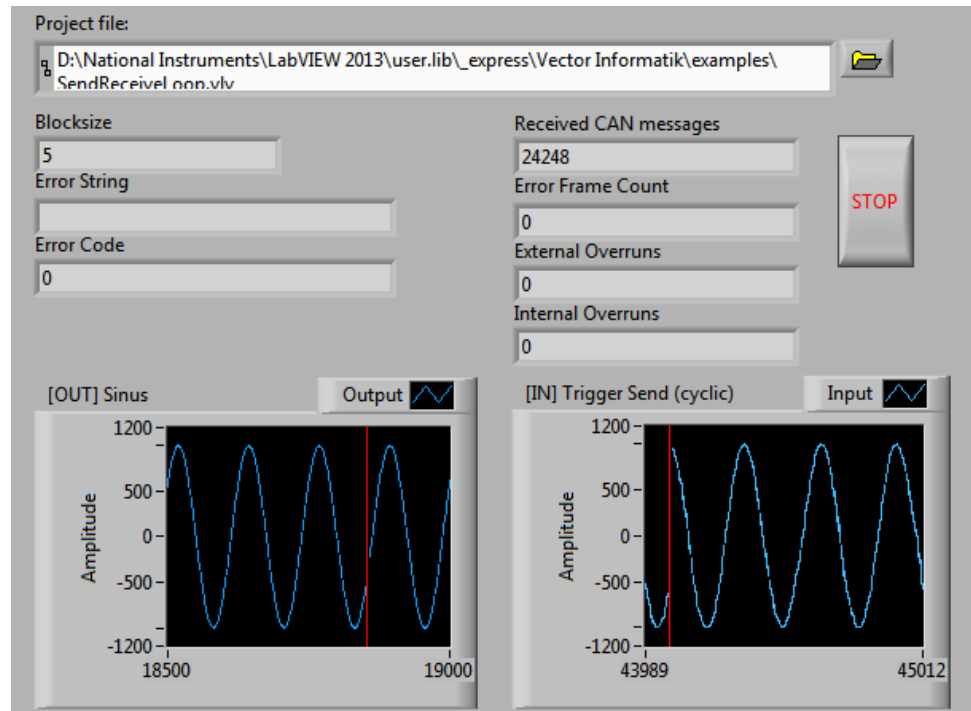


1. Verbinden Sie die beide CAN-Kanäle Ihrer Vector Hardware mit einem geeigneten Kabel. Beachten Sie, dass die Hardware für LabVIEW freigeschaltet sein muss.
2. Öffnen Sie die Vector Hardware Konfiguration (**Start | Einstellungen | Systemsteuerung | Vector Hardware**) und weisen Sie der Vector Hardware die ersten beiden CAN-Kanälen (**LabVIEW CAN1** und **CAN2**) zu.

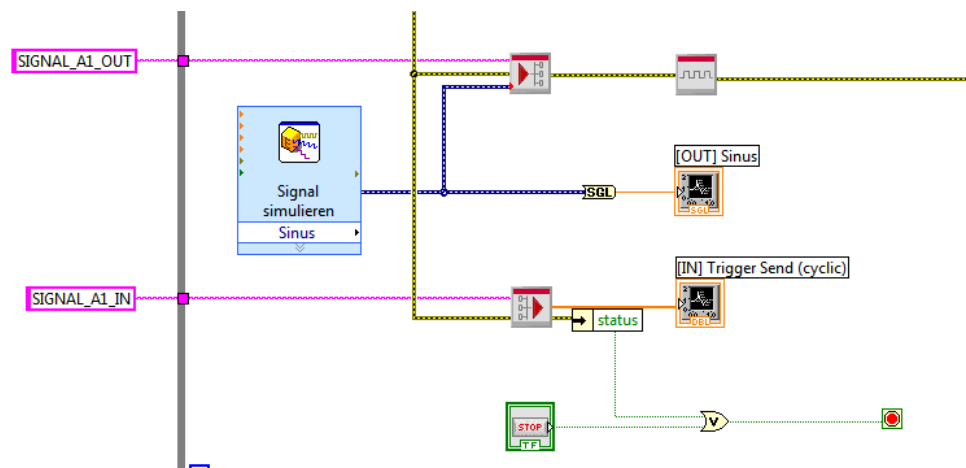


3. Öffnen Sie das VI **SendReceiveLoop.vi** in LabVIEW. Sie können dieses über **Start | Programme | Vector CAN Driver for LabVIEW** aufrufen.
4. Stellen Sie sicher, dass der Pfad (**Project file**) zum Vector LabVIEW Projekt korrekt ist.
5. Starten Sie die Messung mit <STRG+R> oder über das Hauptmenü **Ausführen | Ausführen**.

Zyklische Updates  
mehrer Signale  
(Trigger Send)



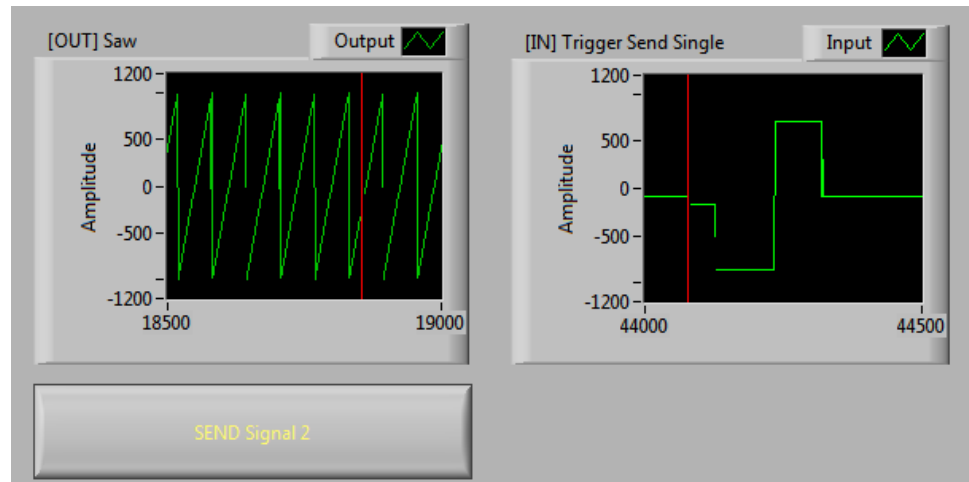
Blockdiagramm



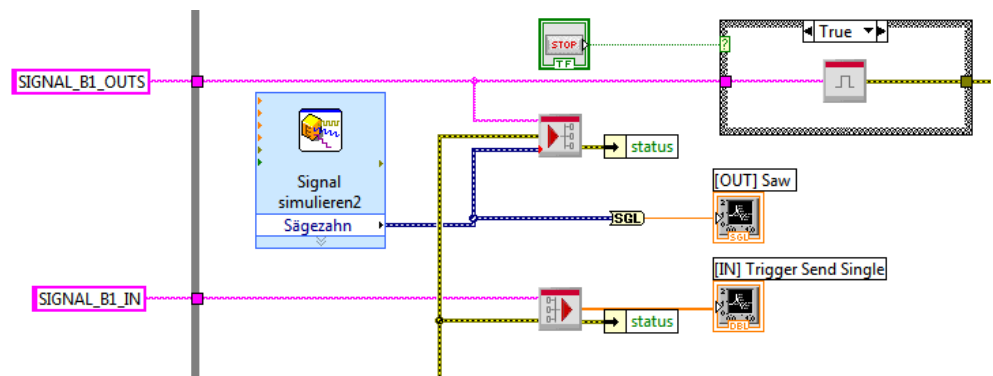
Der Signalverlauf **[OUT] Sinus** zeigt ein von LabVIEW simuliertes Signal, das permanent in den Sendepuffer eines selektierten Signals über das Vector VI **Update Send Buffer** geschrieben wird. Das Versenden des Pufferinhalts auf den CAN-Bus erfolgt mit **Trigger Send** zyklisch in der eingestellten Frequenz und auf dem definierten CAN-Kanal.

Im Signalverlauf **[IN] Trigger Send** ist das von LabVIEW gesendete Sinus-Signal zu sehen, wie er auf dem zweiten CAN-Kanal zyklisch empfangen wird.

Ereignis-getriggerte  
Updates eines  
Signals  
(Trigger Send Single)



Blockdiagramm



Der Signalverlauf **[OUT] Saw** zeigt ein von LabVIEW simuliertes Signal, das permanent in den Sendepuffer eines selektierten Signals über das Vector VI **Update Send Buffer** geschrieben wird. Das Versenden des Pufferinhalts auf den CAN-Bus erfolgt mit Hilfe **Trigger Send Single** manuell über die Schaltfläche **[Send Signal 2]** auf dem definierten CAN-Kanal.

Im Signalverlauf **[IN] Trigger Send Single** ist der aktuelle Wert des Puffers zu sehen, wie er vom Bus empfangen wird.

6. Stoppen Sie die Messung über die Schaltfläche **[Stop]**, um den Vector LabVIEW Treiber korrekt zu schließen und zurückzusetzen.

## 6 Error Codes

In diesem Kapitel finden Sie die folgenden Informationen:

---

6.1 Übersicht

Seite 35

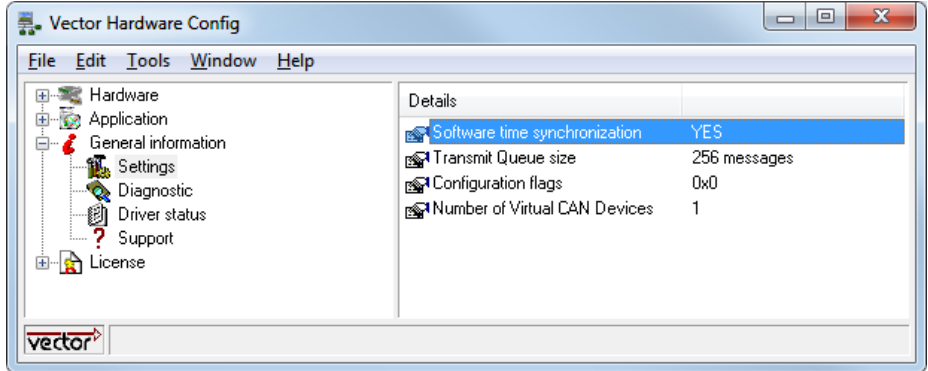
---

## 6.1 Übersicht



**Hinweis:** Für eine Klartextdarstellung zur Laufzeit kann das VI **Error Code To String** genutzt werden (siehe Abschnitt **Error Code to String** auf Seite 22).

Code	Bedeutung
6	DRV_ERR_DIRECTION_NOT_SUPPORTED
9	DRV_ERR_HW_MAPPING
10	DRV_ERR_ACCESS_PROJECTFILE
11	DRV_ERR_LOAD_PROJECTFILE_G
12	DRV_ERR_LOAD_PROJECTFILE_P
13	DRV_ERR_SIGNAL_CONF_LINE_TOO_LONG
14	DRV_ERR_SIGNAL_CONF_PATH2LONG
15	DRV_ERR_SIGNAL_CONF_CANDBACCESS
16	DRV_ERR_SIGNAL_CONF_2MANYCANDB
17	DRV_ERR_SIGNAL_CONF_ADDDBTOGUILIB
18	DRV_ERR_SPLITPATH
19	DRV_ERR_READ_SIGSUBSTRING
20	DRV_ERR_PARAM2LONG
21	DRV_ERR_INVALID_DIRECTION
22	DRV_ERR_INVALID_APPCHANNEL
23	DRV_ERR_APPCHANNEL_NOT_CONFIGURED
24	DRV_ERR_BUS_PARAMETERES_MISSING
25	DRV_ERR_RESOLVE_SIGNAL
26	DRV_ERR_XLGETAPPCONFIG
27	DRV_ERR_BUSCHANNEL_CAPABILITIES
28	DRV_ERR_NO_SIGNAL_CONFIGURED
29	DRV_ERR_OPEN_CAN_PORT
30	DRV_ERR_OPEN_LIN_PORT
31	DRV_ERR_INITACCESS_CAN_PORT
32	DRV_ERR_INITACCESS_LIN_PORT
33	DRV_ERR_SET_CAN_PARAMS
34	DRV_ERR_SET_LIN_PARAMS
35	DRV_ERR_SET_NOTIFICATION
36	DRV_ERR_ACTIVATE_CHANNEL
37	DRV_ERR_SIGNAL_READ_NOT_FOUND
38	DRV_ERR_SIGNAL_BUFFER_NOT_ALLOCATED

39	DRV_ERR_DUPLICATE_OUT_SIGNAL
40	DRV_ERR_LIN_OUT_UNSUPPORTED
41	DRV_ERR_OUT_SIG_NOTFOUND
42	DRV_ERR_COLLECT_SIGNAL_VALUES
43	DRV_ERR_SENDMSG
44	DRV_ERR_OUT_MSG_NOTFOUND
45	DRV_ERR_OUT_MSG_NOTSINGLE
46	DRV_ERR_NO_SIGNALS
47	DRV_ERR_DUPLICATE_SIGNR
48	DRV_ERR_UNKNOWN
50	DRV_ERR_ADD_ACCRANGE
51	DRV_ERR_SET_ACCEPTANCE
52	DRV_ERR_NO_LICENSE
53	DRV_ERR_HW_NOT_SUPPORTED
54	DRV_ERR_CHANNEL_CAPABILITIES <b>Hinweis:</b> Der Fehler 54 wird meist durch die abgeschaltete Software-Zeitsynchronisation hervorgerufen. Klicken Sie auf <b>Start   Einstellungen   Systemsteuerung   Vector Hardware</b> und öffnen die Sektion <b>General information   Settings</b> und schalten <b>Software time synchronization</b> auf <b>YES</b> .
	
55	DRV_ERR_SIGNALALIAS_NOTFOUND
56	DRV_ERR_PROJECT_NOT_OPEN
57	DRV_ERR_SIGNALALIAS_NOTFOUND
58	DRV_ERR_UNKNOWN_PARAM
59	DRV_ERR_BFREQ_LIMIT
60	DRV_ERR_CONFIG_FILE_VERSION
99	DRV_ERR_IN_ERRORCONVERSION
100	DRV_ERR_NO_MEM

## **Get more Information!**

### **Visit our Website for:**

- > News
- > Products
- > Demo Software
- > Support
- > Training Classes
- > Addresses

**[www.vector-worldwide.com](http://www.vector-worldwide.com)**