

# Programmer's Manual Strainbuster ActiveX Control

Version number	1.03
Release date	2002-11-08
Author	A. Cosman



PEEKEL INSTRUMENTS B.V  
INDUSTRIEWEG 161  
3044 AS ROTTERDAM  
TEL: (010)-415 27 22  
FAX: (010)-437 68 26  
EMAIL: [sales@peekel.nl](mailto:sales@peekel.nl)

PEEKEL INSTRUMENTS GMBH  
BERGMANNSTRASSE 43  
44809 BOCHUM  
TEL: 0234/904 1603  
FAX: 0234/904 1605  
EMAIL: [Peekel@t-online.de](mailto:Peekel@t-online.de)

**Contents:**

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Strainbuster Network and Channels .....	4
1.2	Measurement .....	5
1.3	Time labelling .....	5
<b>2</b>	<b>The Interface</b>	<b>6</b>
2.1	Variable types.....	6
2.2	Properties .....	7
2.2.1	General Properties .....	7
	Long ActiveChannelTotal	7
	Integer ChannelCount	7
	Integer CommStatus	7
	Integer DeviceCount	7
	Integer Language	7
	Integer TimeCorrectionMode	7
	Integer WantLogData	7
	Boolean WantTripEvents	8
2.2.2	CAN Interface Properties.....	8
	Long BoardNumber	8
	Long BusSpeed	8
	String DriverInformation	8
	Long HardwareSettings	8
	String InterfaceList	8
	Integer InterfaceType	8
	Integer MaxChannelsPerDevice	8
	Integer MaxDevices	8
	Integer MaxPorts	9
2.2.3	Current Channel Selection Properties.....	9
	Integer DevicePort	9
	Integer DeviceAddress	9
	Integer DeviceChannel	9
2.2.4	Channel Related Properties.....	9
	Boolean BalanceUse	9
	Boolean BalanceUseEx(Integer Port, Integer Address, Integer Channel)	9
	Single BalanceValue	9
	Single BalanceValueEx(Integer Port, Integer Address, Integer Channel)	9
	Single BridgeFactor	9
	Single BridgeFactorEx(Integer Port, Integer Address, Integer Channel)	9
	Integer CalcBalanceUnits	9
	Integer CalcBalanceUnitsEx(Integer Port, Integer Address, Integer Channel)	9
	Single CalcBalanceValue	9
	Single CalcBalanceValueEx(Integer Port, Integer Address, Integer Channel)	9
	Boolean ChannelActive	10
	Boolean ChannelActiveEx(Integer Port, Integer Address, Integer Channel)	10
	Boolean ChannelInUse	10
	Boolean ChannelInUseEx(Integer Port, Integer Address, Integer Channel)	10
	String ChannelName	10
	String ChannelNameEx(Integer Port, Integer Address, Integer Channel)	10
	Integer DeviceType	10
	Integer DeviceTypeEx(Integer Port, Integer Address, Integer Channel)	10
	Integer GainSetting	10
	Integer GainSettingEx(Integer Port, Integer Address, Integer Channel)	10
	Single KFactor	10
	Single KFactorEx(Integer Port, Integer Address, Integer Channel)	10
	Integer MeasurementInterval ( <i>obsolete, use MeasInterval instead</i> )	10
	Integer MeasurementIntervalEx(Integer Port, Integer Address, Integer Channel) ( <i>obsolete, use MeasInterval instead</i> )	10
	Long MeasInterval	10
	Long MeasIntervalEx(Integer Port, Integer Address, Integer Channel)	10

Integer MeasurementType	11
Integer MeasurementTypeEx(Integer Port, Integer Address, Integer Channel)	11
2.2.5 Presentation Measurement Information Properties .....	11
Single MeasValue	11
Single MeasValueEx(Integer Port, Integer Address, Integer Channel)	11
Integer MeasQuality	11
Integer MeasQualityEx(Integer Port, Integer Address, Integer Channel)	11
Integer ChannelUnits	11
Integer ChannelUnitsEx(Integer Port, Integer Address, Integer Channel)	11
String ChannelUnitsText	12
String ChannelUnitsTextEx(Integer Port, Integer Address, Integer Channel)	12
Currency ChannelTimeStamp	12
Currency ChannelTimeStampEx(Integer Port, Integer Address, Integer Channel)	12
Boolean ChannelTripStatus(Integer TripNumber)	12
Boolean ChannelTripStatusEx(Integer Port, Integer Address, Integer Channel, Integer TripNumber)	12
2.3 Methods .....	12
Sub BalanceCentral()	12
Sub BalanceChannels(String ChannelList)	12
Sub BusScan()	12
Sub GetLeadWireResistance(Integer Port, Integer Address, Integer Channel, Single WireLength, Single WireCrossSectionArea, Single MaterialResistance)	12
Sub GetTrip(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName)	13
Sub GetTripWithTimeout(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName, Currency TripTimeout)	13
Sub SelectDefaultInterface()	13
Sub SetDriver(Integer InterfaceType, Long BoardNumber, Long HardwareSettings, Long BusSpeed)	13
Sub SetLeadWireResistance(Integer Port, Integer Address, Integer Channel, Single WireLength, Single WireCrossSectionArea, Single MaterialResistance)	13
Sub SetLocalParam()	13
Sub SendAllParam()	13
Sub SetTrip(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName)	13
Sub SetTripWithTimeout(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName, Currency TripTimeout)	14
Sub ViewProperties()	14
Sub ViewPropertiesEx(String Caption, String ChannelList)	14
2.4 Events .....	14
Event BalanceChanged(Integer Port, Integer Address, Integer Channel, Single Value)	14
Event BusScanComplete()	14
Event CommStatusChanges(Integer Status)	14
Event NewLogValue(Integer Port, Integer Address, Integer Channel, Single Value, Currency msecTimeStamp, unsigned char Quality)	14
Event NewLogValues(Long nValues, Variant Ports, Variant Addresses, Variant Channels, Variant Values, Variant msecTimeStamps, Variant Qualities)	14
Event NewMeasValues()	15
Event NewTripStatus(Integer Port, Integer Address, Integer Channel, Integer TripNumber, Boolean TripActive, Currency msecTimeStamp)	15

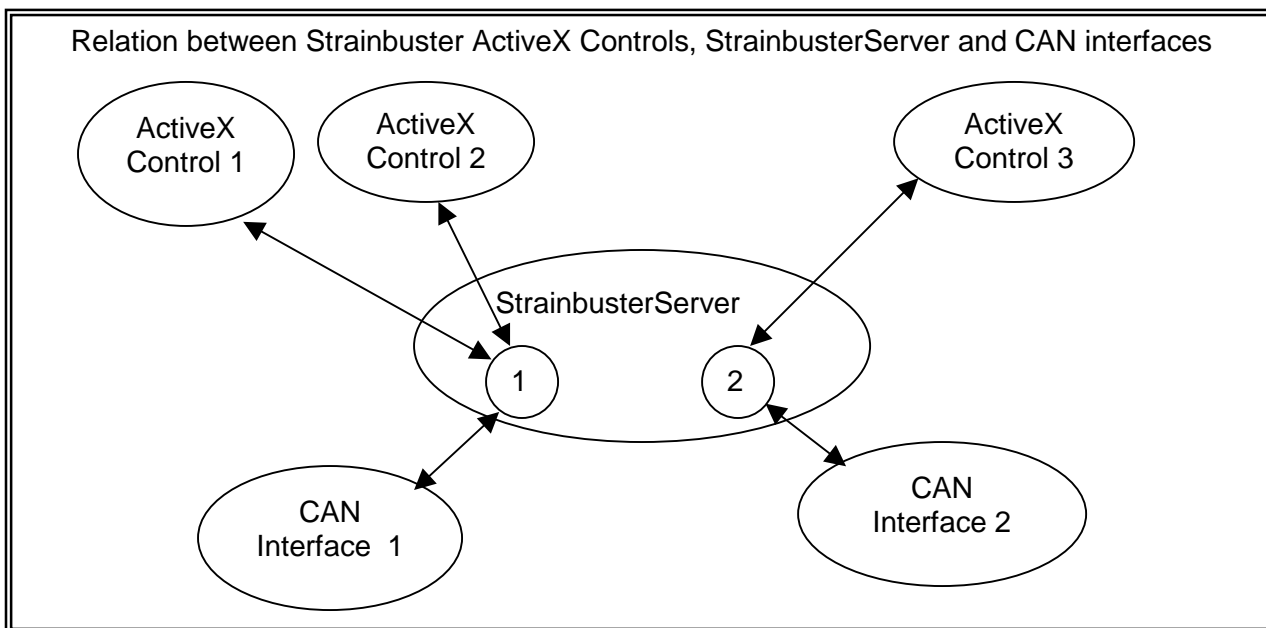
# 1 Introduction

The Strainbuster ActiveX Control contains properties, methods and events to control a collection of strainbusters connected to a single CAN interface. In Visual Basic, it can be used in two ways. As a visible ActiveX component it can be placed on a form. It can also be used as an invisible control by adding a reference to the control in the project and creating a new instance using the 'New' command. In this case, the code would look something like this:

```
Private WithEvents StrainBuster As StrainbusterNet

Private Sub Form_Load()
    Set StrainBuster = New StrainbusterNet
End Sub
```

The strainbuster control communicates with the StrainbusterServer executable, which controls the CAN interface driver(s). The StrainbusterServer can control multiple CAN interfaces at the same time. Each strainbuster control can connect to a single CAN interface through the StrainbusterServer. If several strainbuster controls connect to the same CAN interface, they will share their information through the StrainbusterServer. When one control changes the parameters of the strainbusters, other controls connected to the same CAN interface will reflect these changes.



## 1.1 Strainbuster Network and Channels

The Strainbuster ActiveX control has a number of properties and methods to configure a CAN hardware interface. A driver provided by its manufacturer, which is in turn controlled by the StrainbusterServer, controls each CAN hardware interface. A simple CAN interface has a single connector to connect to a CAN bus. Some CAN interfaces have more than one port and can therefore control multiple CAN buses.

In the ActiveX control a single channel on a strainbuster is therefore addressed by a CAN port number, a strainbuster CAN address and a strainbuster channel number, all of which are numbered from 0 upwards.

Configuration of a single channel can be performed in two ways:

1. Set the properties **DevicePort**, **DeviceAddress**, and **DeviceChannel** to the correct channel and then use properties like **ChannelName** to directly configure a specific channel.

2. Use properties like **ChannelNameEx**, which have three parameters for port, address and channel number.

If you need to set a large number of parameters for several channels, you should first issue a call to **SetLocalParam**, then use **ChannelNameEx** type access to the channels, and finally issue a call to **SendAllParam**. This way, the control will not be updated for every change, making the process a lot faster.

## 1.2 Measurement

To measure values with a strainbuster, two parameters are essential:

**MeasInterval**: you can set a measurement interval in milliseconds with a resolution of 10 milliseconds and minimum of 10 milliseconds, for each channel.

**ChannelActive**: as soon as the ChannelActive property is set to True and the MeasInterval is set at 10 milliseconds or higher, the strainbuster will start to send measurement values at the specified time interval.

The ActiveX control can see these measurement values in two ways:

1. As 'presentation' data in the **MeasValue** property and related properties. Presentation data is updated at most three times per second and provides a way of showing the current measurement value without the overhead of updating it 100 times per second.
2. As 'log' data in the **NewLogValue** or **NewLogValues** event. If you want full access to all measured data, you must first set the **WantLogData** property to an appropriate value. You can choose to get all measurement data as soon as it arrives, or get the data in bursts to reduce the system load.

## 1.3 Time labelling

All measurement values are assigned a time stamp by the StrainbusterServer. Because the strainbusters do not have their own time-of-day clock, the PC-clock is used to determine the time stamp. Depending on your needs, there are two ways in which this time labelling can be performed:

1. Unsynchronised (default): after measurement activation, the time stamp for the first measurement value will be matched to the PC clock. For all subsequent values, the specified measurement interval will be added to the previous time stamp to find the new one. Because the strainbuster timing does not match the PC clock exactly, the time labelling will slowly drift away from the PC clock time, possibly by several seconds per day. Use this mode if you perform short-term measurements or are not interested in synchronising measurement time to PC clock time.
2. Synchronised: operates in the same way as the unsynchronised mode, but differences between the PC clock and the strainbuster time labelling are detected. When the difference becomes too big, 10 milliseconds is added to, or subtracted from, the current strainbuster time stamp. For measurements using a 10 millisecond interval, this means that every once in a while a 'gap' of 20 milliseconds will appear in the time labelling when the PC clock runs faster. When the PC clock runs slower, every once in a while a measurement value will be overwritten with a new measurement value of the same time stamp.

You can set the preferred time labelling mode by using the **TimeCorrectionMode** property.

## 2 The Interface

This chapter describes all properties, methods and events available in the Strainbuster ActiveX Control.

### 2.1 Variable types

In the list of properties, methods and events below, all variable types are indicated using Visual Basic style type names. The table below shows the corresponding types for C/C++ and other languages

Visual Basic	C/C++	Comment
Boolean	BOOL, bool, VARIANT_BOOL	Logical value True or False
Integer	short	2 byte signed integer
Long	long	4 byte signed integer
String	BSTR	string of ASCII characters
Single	float	4 byte single precision IEEE floating-point number
Currency	FILETIME, __int64	8 byte integer

Some background on the use of the Currency type for timestamps:

In Visual Basic, the Currency timestamp as used by the Strainbuster ActiveX Control indicates a UTC-time in milliseconds since the year 1601. To convert this value to a Visual Basic style Date (which is a double precision floating-point number in days since the year 1900), use the following conversion functions:

```
' Windows API support functions for converting time formats
Private Declare Function LocalMSecTimeToMSecTime Lib "kernel32" Alias
    "LocalFileTimeToFileTime" (lpLocalFileTime As Currency, lpMSecTime As
    Currency) As Long
Private Declare Function MSecTimeToLocalMSecTime Lib "kernel32" Alias
    "FileTimeToLocalFileTime" (lpMSecTime As Currency, lpLocalMSecTime As
    Currency) As Long

' Convert a Visual Basic style Date/Time, local time
' to a time in milliseconds since 1-1-1601, UTC
Private Function DateToMSecTime(ByVal dat As Date) As Currency
    Dim msecUTC As Currency, msecLocal As Currency
    msecLocal = CCur((CDBl(dat) + 109205) * 86400000# + 0.9)
    Call LocalMSecTimeToMSecTime(msecLocal, msecUTC)
    DateToMSecTime = msecUTC
End Function

' Convert a time in milliseconds since 1-1-1601, UTC
' to a Visual Basic style Date/Time, local time
Private Function MSecTimeToDate(ByVal msec As Currency) As Date
    Dim msecLocal As Currency
    Call MSecTimeToLocalMSecTime(msec, msecLocal)
    MSecTimeToDate = (CDBl(msecLocal) / 86400000# - 109205#)
End Function
```

In Visual C and others, the 8 byte integer timestamp corresponds to a FILETIME as returned by the Windows API function CoFileTimeNow, which is a UTC-time in 100 nanosecond intervals since the

year 1601. Windows API functions like `FileTimeToLocalFileTime` and `FileTimeToSystemTime` can help to convert this number to a presentable local time.

## 2.2 Properties

### 2.2.1 General Properties

#### **Long ActiveChannelTotal**

Read-only, indicates the total amount of active channels over all currently connected devices.

#### **Integer ChannelCount**

Read-only, indicates the amount of available channels for the current device. Whether or not a channel is available is determined by the '**ChannelInUse**' property.

#### **Integer CommStatus**

Read-only, current CAN interface communication status. The status contains 3 bits:

bit 0: 1 = CAN interface connection established, 0 = no CAN interface connection

bit 1: 2 = One or more devices are not responding, 0 = all devices respond as expected

bit 2: 4 = One or more devices are not able to send all measurements (bus load too high), 0 = All measurement data arrives as expected

Available constants:

**sbcNoInterfaceConnection** = 0, "No connection to a CAN interface exists"

**sbcOK** = 1, "CAN interface connection OK (bit 0 set)"

**sbcDisconnectedDevices** = 3, "CAN interface connection OK (bit 0 set) but one or more devices do not respond (bit 1 set)"

**sbcMissingValues** = 5, "CAN interface connection OK (bit 0 set) but one or more devices have missing values (bit 2 set)"

#### **Integer DeviceCount**

Read-only, indicates the amount of currently connected devices.

#### **Integer Language**

Language used in the user interface. At startup, the default language is determined from the Windows regional settings. Available values:

**sblnDefault** = 0, "Language not set, default = English"

**sblnGerman** = 7, "Deutsch"

**sblnEnglish** = 9, "English"

**sblnDutch** = 19, "Nederlands"

#### **Integer TimeCorrectionMode**

Determines if the measurement timestamps should be kept in sync with the PC clock. Available values:

**sbtcNever** = 0, "Never adapt timestamps to match PC clock time"

**sbtcAlways** = 1, "Always keep timestamps in sync with PC clock time"

**sbtcDatasharerNotLogging** = 2, "Only timestamps in sync with PC clock time when Datasharer logging is inactive"

#### **Integer WantLogData**

Indicates whether or not all measurement data should be available to the ActiveX control. Available values:

**sbldNone** = 0, "Don't receive full measurement data, only presentation values 3 times per second"  
**sbldBurstMode** = 1, "Receive full measurement data in bursts (low processor load)"  
**sbldContinuousMode** = 2, "Receive full measurement data as soon as possible (high processor load)"

### Boolean WantTripEvents

Indicates whether or not trip information should be available. If True, then any trip activation will raise a '**NewTripStatus**' event. If False, no trip information will be available for this channel.

## 2.2.2 CAN Interface Properties

### Long BoardNumber

CAN Interface board number. The default value 0 should only be changed when there are several CAN interfaces of the same type available.

### Long BusSpeed

CAN Interface bus speed in bps. Example: 10000 = 10 kbps, 1000000 = 1 Mbps. Unacceptable speeds are automatically 'rounded' to the closest valid speed. Suitable values are 10000, 20000, 50000, 125000, 250000, 500000, 800000 and 1000000.

### String DriverInformation

Read-only text string containing CAN interface device driver information. This may include interface parameters, manufacturer and driver version number.

### Long HardwareSettings

CAN Interface hardware parameters. For the Peak CAN Dongle, this parameter contains the I/O address and IRQ of the parallel port, encoded as 'IOAddress \* 65536 + IRQ'. For LPT1, this could be &H03780007. Other CAN interface types currently have no specific HardwareSettings and ignore this value.

### String InterfaceList

Read-only, textstring describing all available CAN interface types. The string is made up of fields, separated by a '|' character. Each interface is described by five fields.

Field 1: Interface driver DLL name.

Field 2: Interface driver description (text for presentation).

Field 3: Number of CAN interface ports available on this type of interface.

Field 4 & 5: Interface dependant additional numeric information.

### Integer InterfaceType

CAN Interface type. The integer value relates to the list of available CAN interfaces, property '**InterfaceList**'. 0 = No interface selected, 1 = First interface in the **InterfaceList**, and so on.

### Integer MaxChannelsPerDevice

Read-only, indicates the maximum number of channels per device. Devices can have channel numbers between 0 and (MaxChannelsPerDevice-1).

### Integer MaxDevices

Read-only, indicates the device address range. Devices can have addresses between 0 and (MaxDevices-1).

**Integer MaxPorts**

Read-only, indicates the number of CAN interface ports on the current CAN hardware. Allowable port numbers are between 0 and (MaxPorts-1).

## 2.2.3 Current Channel Selection Properties

**Integer DevicePort**

CAN Interface port for selection of the current channel. This value is 0 unless the selected CAN hardware has more than 1 CAN interface. Together with **DeviceAddress** and **DeviceChannel** this property determines the current channel. Many other properties affect the channel selected by these properties.

**Integer DeviceAddress**

Strainbuster device address for selection of the current channel. Together with **DevicePort** and **DeviceChannel** this property determines the current channel. Many other properties affect the channel selected by these properties.

**Integer DeviceChannel**

Strainbuster channel for selection of the current channel. Together with **DevicePort** and **DeviceAddress** this property determines the current channel. Many other properties affect the channel selected by these properties.

## 2.2.4 Channel Related Properties

**Boolean BalanceUse****Boolean BalanceUseEx(Integer Port, Integer Address, Integer Channel)**

Indicates whether the balance value is used for the channel.

**Single BalanceValue****Single BalanceValueEx(Integer Port, Integer Address, Integer Channel)**

Balance value for the channel, in V (Volts) for DC measurements, in  $\Omega$  (Ohms) for resistance-based measurements (i.e. strain and Pt-100).

**Single BridgeFactor****Single BridgeFactorEx(Integer Port, Integer Address, Integer Channel)**

Bridge factor for strain measurement for the channel.

**Integer CalcBalanceUnits****Integer CalcBalanceUnitsEx(Integer Port, Integer Address, Integer Channel)**

Read-only, units for the calculated balance value. Corresponds to **ChannelUnits**, except when user-defined units are used. See **ChannelUnits** property for more information and available values.

**Single CalcBalanceValue****Single CalcBalanceValueEx(Integer Port, Integer Address, Integer Channel)**

Balance value in measurement units for the channel.

**Boolean ChannelActive****Boolean ChannelActiveEx(Integer Port, Integer Address, Integer Channel)**

Indicates whether measurement is active for the channel. An active channel produces measurement values using the selected **MeasInterval**, an inactive channel does not produce measurement values.

**Boolean ChannelInUse****Boolean ChannelInUseEx(Integer Port, Integer Address, Integer Channel)**

Indicates whether the channel is available.

**Single ChannelLinearizationFactor(Integer FactorNumber)****Single ChannelLinearizationFactor(Integer Port, Integer Address, Integer Channel, Integer FactorNumber)**

Indicates the linearization factor for a channel. The formula used to convert measurement units to physical units:

$$\text{PhysUnits} = \text{ChannelLinearizationFactor}(1) * \text{MeasUnits} + \text{ChannelLinearizationFactor}(0)$$
**String ChannelName****String ChannelNameEx(Integer Port, Integer Address, Integer Channel)**

User defined name for the channel. The format of the name is restricted. Illegal names are automatically transformed to legal names.

**Integer DeviceType****Integer DeviceTypeEx(Integer Port, Integer Address, Integer Channel)**

Type of strainbuster device. Currently always 1.

**Integer GainSetting****Integer GainSettingEx(Integer Port, Integer Address, Integer Channel)**

Gain selection for the channel. Available values:

**sbgnGain2\_5mV** = 0, "Gain +/-2.5 mV, or 1 mV/V, or +/-5000  $\mu\text{m}/\text{m}$ , or +/-5  $^{\circ}\text{C}$ "

**sbgnGain10mV** = 1, "Gain +/-10 mV, or 4 mV/V, or +/-20000  $\mu\text{m}/\text{m}$ , or +/-20  $^{\circ}\text{C}$ "

**sbgnGain60mV** = 2, "Gain -50 - +60 mV, or 24 mV/V, or +/-100000  $\mu\text{m}/\text{m}$ , or -90 - +110  $^{\circ}\text{C}$ "

**sbgnGain180mV** = 3, "Gain -100 - +180 mV, or 72 mV/V, or +/-220000  $\mu\text{m}/\text{m}$ , or -100 - +300  $^{\circ}\text{C}$ "

**Single KFactor****Single KFactorEx(Integer Port, Integer Address, Integer Channel)**

k-factor for strain measurement for the channel.

**Integer MeasurementInterval (obsolete, use MeasInterval instead)****Integer MeasurementIntervalEx(Integer Port, Integer Address, Integer Channel) (obsolete, use MeasInterval instead)****Long MeasInterval****Long MeasIntervalEx(Integer Port, Integer Address, Integer Channel)**

Measurement interval in milliseconds for the channel. For values of 10 milliseconds and higher in 10 milliseconds increments, the strainbuster will send measurement values to the PC using the specified interval, if 'ChannelActive' is True.

## Integer MeasurementType

### Integer MeasurementTypeEx(Integer Port, Integer Address, Integer Channel)

Type of measurement for the channel. Available values:

**sbmtNone** = 0, "Measurement type not set"  
**sbmtStrain350** = 1, "350 Ohms 1/4 bridge strain measurement"  
**sbmtPT100** = 2, "Pt-100 temperature measurement"  
**sbmtStrain120** = 3, "120 Ohms 1/4 bridge strain measurement"  
**sbmtDC** = 4, "DC voltage measurement"  
**sbmtStrainHalfBridge** = 5, "1/2 bridge strain measurement"  
**sbmtTransducerFull** = 20, "Transducer 1/1 bridge measurement"  
**sbmtTransducerHalf** = 21, "Transducer 1/2 bridge measurement"  
**sbmtStrainFullBridge** = 36, "1/1 bridge strain measurement"

## 2.2.5 Presentation Measurement Information Properties

### Single MeasValue

#### Single MeasValueEx(Integer Port, Integer Address, Integer Channel)

Read-only, measurement value for the channel. Depending on the current state of the channel, this value may be old or invalid. Check the 'MeasQuality' property to determine the status of the value. This value is updated at most about three times per second, and is meant for presentation, not logging.

### Integer MeasQuality

#### Integer MeasQualityEx(Integer Port, Integer Address, Integer Channel)

Read-only, measurement quality indication for the channel. The quality indication is based on OPC quality fields. (OPC = OLE for Process Control, see also [www.opcfoundation.org](http://www.opcfoundation.org)) This value is updated at most about three times per second, and is meant to be used for presentation, not logging. The following values are used:

**sbquDisconnect** = 11, "No connection to the device, value is constant"  
**sbquNoRecentData** = 28, "Device is not active, measurement value is old"  
**sbquTooLow** = 85, "Device is out of range (below range)"  
**sbquTooHigh** = 86, "Device is out of range (above range)"  
**sbquMissingValues** = 88, "Device is missing values (some values before this one were lost)"  
**sbquOK** = 192, "Device measurement status is OK"

### Integer ChannelUnits

#### Integer ChannelUnitsEx(Integer Port, Integer Address, Integer Channel)

Read-only, units for the channel. Available values:

**sbunNone** = 0, "Units not set"  
**sbunVolts** = 1, "Voltage (V)"  
**sbunMeterPerMeter** = 2, "Meter per meter (m/m)"  
**sbunDegreesCelsius** = 3, "Degrees Celsius (°C)"  
**sbunVoltPerVolt** = 4, "Volt per volt (V/V)"  
**sbunUserDefined** = 99, "User defined"

**String ChannelUnitsText****String ChannelUnitsTextEx(Integer Port, Integer Address, Integer Channel)**

Gets/sets the units for the measurement value as text. When using user-defined units (see **ChannelUnits** property), this property can be set to any suitable units. For predefined units, this property shows the units in textual form.

**Currency ChannelTimeStamp****Currency ChannelTimeStampEx(Integer Port, Integer Address, Integer Channel)**

Read-only, latest measurement time for the channel. This value is updated at most about three times per second, and is meant for presentation, not logging. The timestamp indicates a UTC time in milliseconds since 1-1-1601, compatible with the Windows API calls CoFileTimeNow or GetSystemTimeAsFileTime.

**Boolean ChannelTripStatus(Integer TripNumber)****Boolean ChannelTripStatusEx(Integer Port, Integer Address, Integer Channel, Integer TripNumber)**

Read-only, current trip status for a trip on a channel. This value will not be valid unless the 'WantTripEvents' property is set to True.

## 2.3 Methods

**Sub BalanceCentral()**

Perform a balance command on all connected channels. This will not have any effect on Pt-100 channels.

**Sub BalanceChannels(String ChannelList)**

Perform a balance command on a selection of connected channels. ChannelList should contain a comma-separated list of port, address and channelnumber for all channels to be balanced. Eg. ChannelList = '0,2,0,0,2,1' will balance channels 0 and 1 of device address 2 on CAN port 0.

**Sub BusScan()**

Issues a broadcast on the CAN bus to find connected devices and their channels. When devices respond to this broadcast, their available channels are marked available ('ChannelInUse' = True). This command returns immediately, a **BusScanComplete** event is raised when all device responses have been received. Note that any channels already marked available but not detected during the scan will not be removed.

**Sub GetLeadWireResistance(Integer Port, Integer Address, Integer Channel, Single WireLength, Single WireCrossSectionArea, Single MaterialResistance)**

Retrieve the lead wire resistance information for a channel. The WireLength is specified in meters, the WireCrossSectionArea in mm<sup>2</sup>, the MaterialResistance in μOhms-cm. The actual lead wire resistance in Ohms, is calculated by:  $(MaterialResistance * WireLength) / (100 * WireCrossSectionArea)$ .

**Sub GetTrip(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName)**

Retrieves the trip configuration for a specific trip on a channel. Port, Address, Channel and Trip are input values, TripType, Level, Hysteresis and TripName are output values. Obsolete, for new applications: use **GetTripWithTimeout**. See also '**GetTripWithTimeout**', '**SetTrip**', '**SetTripWithTimeout**'.

**Sub GetTripWithTimeout(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName, Currency TripTimeout)**

Retrieves the trip configuration for a specific trip on a channel. Port, Address, Channel and Trip are input values, TripType, Level, Hysteresis, TripName and TripTimeout are output values. See also '**SetTripWithTimeout**'.

**Sub SelectDefaultInterface()**

Selects the default CAN interface (first valid interface configured on the PC). Each time a valid connection to a CAN interface is established, the StrainbusterServer will store the associated settings in the Windows registry. This call retrieves the settings for the first interface configuration stored in this way.

**Sub SetDriver(Integer InterfaceType, Long BoardNumber, Long HardwareSettings, Long BusSpeed)**

Select a specific CAN interface. For a description of the parameters, see the properties with corresponding names.

**Sub SetLeadWireResistance(Integer Port, Integer Address, Integer Channel, Single WireLength, Single WireCrossSectionArea, Single MaterialResistance)**

Configures the lead wire resistance information for a channel. The WireLength is specified in meters, the WireCrossSectionArea in mm<sup>2</sup>, the MaterialResistance in µOhms-cm. The actual lead wire resistance in Ohms, is calculated by:  $(MaterialResistance * WireLength) / (100 * WireCrossSectionArea)$ .

**Sub SetLocalParam()**

Suspend immediate update of channel configuration parameters. This command blocks the direct processing of changes in the various properties. Use this command when changing settings for a large number of channels to speed up processing. See '**SendAllParam**' for further information.

**Sub SendAllParam()**

Send all suspended channel configuration parameters to the driver. If you need to make a large number of channel configuration calls, you should call **SetLocalParam** first, then configure the channels, and finally call **SendAllParam**. This will suppress the immediate update of the ActiveX control and the StrainbusterServer and dramatically increases the speed at which the changes are processed.

**Sub SetTrip(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName)**

Configures a trip on a channel. You can configure up to four individual trips on a channel, numbered from 0 up to 3. TripName is a free-formed description of the trip. TripType can have the following values:

**sbttDisabled** = 0, "Trip disabled (unused)"

**sbttOverflow** = 1, "Trip on overflow"

**sbttUnderflow** = 2, "Trip on underflow"

**Sub SetTripWithTimeout(Integer Port, Integer Address, Integer Channel, Integer Trip, Integer TripType, Single Level, Single Hysteresis, String TripName, Currency TripTimeout)**

Configures a trip on a channel. You can configure up to four individual trips on a channel, numbered from 0 up to 3. TripName is a free-formed description of the trip. TripType can have the following values:

**sbttDisabled** = 0, "Trip disabled (unused)"

**sbttOverflow** = 1, "Trip on overflow"

**sbttUnderflow** = 2, "Trip on underflow"

**sbttRetrigOverflow** = 3, "Trip on overflow, retriggerable"

**sbttRetrigUnderflow** = 4, "Trip on underflow, retriggerable"

**Sub ViewProperties()**

Show the configuration property pages.

**Sub ViewPropertiesEx(String Caption, String ChannelList)**

Show the configuration property pages for a specific channel selection. 'Caption' specifies the windowname to use, 'ChannelList' specified a list of channels to configure. If 'ChannelList' is empty, the full configuration dialog is shown. If it contains a list of channels (see **BalanceChannels** for a description of the format), the dialog only shows the channel and trips configuration pages and preselects the channels specified.

## 2.4 Events

**Event BalanceChanged(Integer Port, Integer Address, Integer Channel, Single Value)**

Raised when balance value changes for a channel. The Value parameter contains the new balance value in V (Volts).

**Event BusScanComplete()**

Raised when a **BusScan** completes.

**Event CommStatusChanges(Integer Status)**

Raised when the communication status of the CAN interface changes. See the property '**CommStatus**' for a description of the Status parameter.

**Event NewLogValue(Integer Port, Integer Address, Integer Channel, Single Value, Currency msecTimeStamp, unsigned char Quality)**

Raised when new measurements arrive for a channel. This event will only be raised when the **WantLogData** property is set to **sbldContinuousMode** = 2. An event will be raised for each new measurement value for each active channel. When using large amounts of channels or high measurement speeds, this will generate a high processor load. Use '**sbldBurstMode**' whenever possible to reduce this load.

**Event NewLogValues(Long nValues, Variant Ports, Variant Addresses, Variant Channels, Variant Values, Variant msecTimeStamps, Variant Qualities)**

Raised when new batch of measurements arrives, at most about three times per second. This event will only be raised when the **WantLogData** property is set to **sbldBurstMode** = 1.

The Variant parameters contain arrays with indexes from 0 up to (nValues-1). The types of the values in the arrays are:

Ports: Array of Integer

Addresses: Array of Integer

Channels: Array of Integer

Values: Array of Single

msecTimeStamps: Array of Currency, see also property **ChannelTimeStamp**

Qualities: Array of Byte, see also property **MeasQuality**

#### **Event NewMeasValues()**

Raised when the presentation measurement values are updated, which happens at most about three times per second. When this event is raised, new information can be found in '**MeasValue**' and associated properties.

#### **Event NewTripStatus(Integer Port, Integer Address, Integer Channel, Integer TripNumber, Boolean TripActive, Currency msecTimeStamp)**

Raised when a trip status changes. This event will only be raised when the **WantTripEvents** property is set to True.

TripNumber specifies the trip (between 0 and 3), TripActive = True when the trip is active, msecTimeStamp contains the time at which the trip event occurred (see also property ChannelTimeStamp).