JBC

www.jbctools.com

Connect Library

Programmer's Guide

0011580/1216

DLL Version: 3.19.1.7

Contents

1 Introduction

The JBC Connect Library provides complete functionality to monitor and control JBC stations. In order to use this library and, in a more general view, to connect JBC stations to a PC, the proper USB to UART bridge drivers must be installed. This drivers are provided with the library distribution. See "Installing USB driver for the new Excellence series stations" in page 51 for installing the driver for the new Excellence series.

The library has been developed using Microsoft Visual Studio 2010 and is provided as a Dynamic Link Library (DLL); this means that it only can be used in Windows OS. The library is implemented in VisualBasic using **.NET Framework 4.0**.

The library (DLL), two test applications with source code and the drivers for the USB to UART bridge are included in the library distribution set.

1.1 System Requirements

Operating System: Windows XP / Vista / 7 / 8 / 8.1 / 10

Processor: Intel i3

RAM: at least 1GB, 2GB recommended

Hard disk: 2MB available hard disk space

NET Framework 4.0 installed

1.2 Supported Stations

The JBC stations supported by the library are listed below. The minimum program version needed is showed for each station model.

Station	Minimum program version needed
DM	9996761
DD	9996762
DDR	9996779
DI	9996763
HD	9996764
HDR	9996780
CD/CF *	9996824
CS/CV	9996766
CP	9996767
NA	9996768
DDE, HDE, DME, NAE	All
JTSE	All

2 Library Description

The library is composed by one main class and two tool classes. The main class is called **JBC_API**. Inside this class are all the public constants, data structures, events and functions required by the programmer or, in other words, served by the library. The other two classes are **Ctemperature**, to store a temperature, and **Cerror**, to indicate errors to the user. These three classes are stored in the namespace **JBC_Connect**.

2.1 JBC_API Class

This is the main class of the library. It must be created in order to use the library. In this class, constant lists (enumerations), data structures (structure), events and methods are found:

- ***Constant lists**:* These lists of constants are used as a variable type for some of the parameters of the methods in the class. For example, a list of possible station tools. They are listed in "Constants" on page 9.
- ***Data structures**:* There are some data structures defined to be used as a parameter of some methods, typically the continuous mode methods. They are listed in "Structures" on page 9.
- ***Events**:* There are three events in the class. Two of them are used to detect the connection and disconnection of JBC stations to the PC and the third one is used to pass errors (using the Cerror class previously indicated). See "Events" on page 11.
- ***Methods**:* There are several methods in this class. Those methods are the API and are designed in a C-style library, usually starting with "Get" or "Set".

2.1.1 Create

First of all, create the JBC_API class.

```
jbc_api = New JBC_Connect.JBC_API()
```

2.1.2 Start

Then, you must call jbc_api.StartSearch() to start searching for connected stations.

2.1.3 Close

Before close the program, call jbc_api.Close() to free all the resources and threads.

2.1.4 Data and functions supported by station types

Methods and data are marked with the following icons

 Supported by soldering stations only

 Supported by desoldering stations only

No icon indicates that are supported by both types of stations

2.1.5 Constants

- ***ControlModeConnection:*** Control mode status: MONITOR or CONTROL.
- ***CounterTypes:*** Types of counters: GLOBAL_COUNTER or PARTIAL_COUNTER
- ***GenericStationTools****:* List of general tool models.
- ***Port****:* General list of station ports.
- ***PortWorkMode_HA****:* List of work modes. 
- ***PedalAction****:* List of pedal actions. 
- ***OnOff****:* Used in some methods as ON or OFF status value.
- ***SpeedContinuousMode****:* List of available speeds for the continuous mode.
- ***StationError****:* List of internal station errors.
- ***eStationType:*** List of station types, like SOLD (soldering stations) or HA (Hot Air stations).
- ***CTemperature.TemperatureUnit****:* List of temperature units available for the station.
- ***ToolError****:* List of internal station port errors.
- ***ToolFutureMode****:* List of modes automatically reached by the station. 
- ***ToolStatus:*** List of the current status of the tool. 
- ***ToolStatus_HA:*** List of the current status of the tool. 
- ***ToolTemperatureLevels****:* List of temperature levels available for the tools.
- ***ToolTimeHibernation****:* List of possible hibernation delay times. 
- ***ToolTimeSleep****:* List of possible sleep delay times. 

2.1.6 Structures

CToolStartMode_HA*:* Defines the start mode for hot air desoldering stations.

- **ToolButton**: OnOff
- **StandOut**: OnOff
- **Pedal**: PedalAction: NONE, PULL, PULSE, PUSH

CContinuousModeStatus*:* Defines the status of the continuous mode, with selected ports and speed.

1. **port1**: used to indicate if port 1 is active in continuous mode.
2. **port2**: used to indicate if port 2 is active in continuous mode.
3. **port3**: used to indicate if port 3 is active in continuous mode.
4. **port4**: used to indicate if port 4 is active in continuous mode.
5. **speed**: used to indicate the continuous mode speed (period).

***stContinuousModePort**:* Defines the data received from a port in continuous mode, for soldering stations.

1. **port***:* used to indicate from which port this data is coming.
2. **temperature***:* used to pass the port current tool tip temperature.
3. **power***:* used to pass the port current tool power consumption in per thousand units.
4. **status***:* used to pass the current port tool status.

***stContinuousModeData**:* Defines the data returned by the continuous mode, for soldering stations.

1. **data***:* a vector of stContinuousModePort structures that contains the information reported by the station for all the ports activated in the continuous mode.
2. **sequence***:* used to pass the corresponding sequence ordering value for the current data transmission.

***stContinuousModePort_HA**:* Defines the data received from a port in continuous mode, for hot air desoldering stations.

3. **port***:* used to indicate from which port this data is coming.
4. **temperature***:* used to pass the port current tool air temperature
5. **flow:** used to pass the port current tool air flow, in per thousand units.
6. **power***:* used to pass the port current tool power consumption in per thousand units.
7. **externalTC1_Temp***:* used to pass the port current external TC 1 air temperature.
8. **externalTC2_Temp***:* used to pass the port current external TC 2 air temperature.
9. **timeToStop***:* used to pass the port current time to stop, in seconds.
10. **status***:* used to pass the current port tool status.

***stContinuousModeData_HA**:* Defines the data returned by the continuous mode, for hot air desoldering stations.

1. **data***:* a vector of stContinuousModePort_HA structures that contains the information reported by the station for all the ports activated in the continuous mode.
2. **sequence***:* used to pass the corresponding sequence ordering value for the current data transmission.

CFeaturesData: features supported by the station. You should query these data to know if a function is supported by the station. For example, some stations do not allow setting the "beep" value, so see "DisplaySettings" feature to know if you can call "GetStationBeep" and "SetStationBeep" functions.

- Alarms (Boolean): supports functions related with alarms.
- AllToolsSamePortSettings (Boolean): tools settings are the same for all tools.
- Cartridges (Boolean): supports functions related with cartridges.
- DelayWithStatus (Boolean): station allows you to activate and deactivates tool sleep setting maintaining the sleep value (SetPortToolSleepDelayEnabled)
- DisplaySettings (Boolean): true = supports TempUnits, N2Mode, HelpText and Beep functions.
- Ethernet (Boolean): station supports Ethernet connection.
- FirmwareUpdate (Boolean): station supports firmware update through this API.
- MaxTemp (CTemperature): maximum temperature supported by the station.
- MinTemp (CTemperature): minimum temperature supported by the station.
- MaxPowerLimit (Integer): maximum power supported by the station.
- PartialCounters (Boolean): station supports partial counters.
- Peripherals (Boolean): station permits peripherals.
- Robot (Boolean): station supports robot control.
- SubStations (Boolean): station supports substations connected to it.
- TempLevelsWithStatus (Boolean): fix temperature not supported, and temperature levels settings allows you to activate and deactivates a level maintaining the temperature value (SetPortToolTempLevelEnabled).
- TempLevels (Boolean): supports temperature levels.

2.1.7 Events

NewStationConnected:

- *Description:* This event is launched when a new station has been connected to the PC.

- *Prototype:* Public **Event** NewStationConnected(ByVal stationID as **Long**)

- *Params:* **stationID** – The identifier of the detected station. User must use this ID in order to use the API methods to manage the station.

StationDisconnected*:*

- *Description:* This event is launched when a station has been disconnected from the PC.

- *Prototype:* Public **Event** StationDisconnected (ByVal stationID as **Long)**

- *Params:* **stationID** – The identifier of the disconnected station. User must not use this ID anymore.

UserError*:*

- *Description:* This event is launched when a method detects an error on its input parameters. User must use this event in order to catch these errors. In the methods descriptions that follow these errors are specified.

- *Prototype:* Public **Event** UserError(ByVal stationID as **Long**, ByVal err as **Cerror)**

- *Params:* **stationID** – The identifier of the station.

  err – The error occurred as a Cerror object. See Cerror class description for more information.

TransactionFinished*:*

- *Description:* This event is launched when the library receives the response for the SetTransaction function. You can use this function to know that all previous functions have been processed by the station.

- *Prototype:* Public **Event** TransactionFinished(ByVal stationID as **Long** , ByVal transactionID as **UInteger)**

- *Params:* **stationID** – The identifier of the station.

transactionID – The transaction number returned by SetTransaction method.

2.1.8 Methods

Following is the list of all the methods for this class that is that main one. Each of these methods is described with five parameters:

- Description: General method behaviour description.
- Prototype: VisualBasic method prototype code.
- Parameters: Explanation of the parameters of the method.
- Returns: If the method is a function type that returns a value here is described this value.

For the possible errors thrown by the event "UserError" as Cerror objects for the method, see "Cerror Class" on page 49.

An important thing to note is that when a user error is thrown by the event those methods that return values will return a non valid value.

2.1.9 Station Info Methods

2.1.9.1 GetStationType

Description: Returns the station type of the indicated station: SOLD (soldering stations) or HA (hot air desoldering stations).

Prototype: Public **function** GetStationType(ByVal stationID as **Long)** as **eStationType**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The type of the station. Ex: SOLD

2.1.9.2 GetStationCOM

Description: Returns the COM port name of the indicated station, or the station address if connection type is Ethernet.

Prototype: Public **function** GetStationCOM(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The COM port name or Ethernet address of the station. Ex: "COM5"

2.1.9.3 GetStationConnectionType

Description: Returns U (USB) or E (Ethernet).

Prototype: Public **function** GetStationConnectionType(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: U (USB) or E (Ethernet).

2.1.9.4 GetStationProtocol

Description: Returns the communications protocol version of the indicated station.

Prototype: Public **function** GetStationProtocol(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The communications protocol version of the station. Ex: "02"

2.1.9.5 GetStationModel

Description: Returns the model name of the indicated station.

Prototype: Public **function** GetStationModel(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The model name of the station.

2.1.9.6 GetStationModelType

Description: Returns the model type of the indicated station.

Prototype: Public **function** GetStationModel(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The model type of the station.

2.1.9.7 GetStationModelVersion

Description: Returns the model version of the indicated station.

Prototype: Public **function** GetStationModel(ByVal stationID as **Long)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The model version of the station.

2.1.9.8 GetStationHWversion

Description: Returns the hardware version of the indicated station.

Prototype: Public **function** GetStationHWversion(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The hardware version of the station.

2.1.9.9 GetStationSWversion

Description: Returns the software version of the indicated station.

Prototype: Public **function** GetStationSWversion(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The software version of the station.

2.1.9.10 GetPortCount

Description: Returns the number of ports of the indicated station. It depends on the model.

Prototype: Public **function** GetPortCount(ByVal stationID as **Long)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The number of ports of the station.

2.1.9.11 GetStationTools

Description: Returns the list of supported tools of the indicated station. It depends on the model.

Prototype: Public **function** GetStationTools(ByVal stationID as **Long)** as **GenericStationTools()**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: Supported tools of the station.

2.1.9.12 GetStationFeatures

Description: Returns the features supported by the station. It depends on the model, model type and model version. Query the features of the station to know if methods are supported by the station.

Prototype: Public **function** GetStationFeatures(ByVal stationID as **Long)** as **CFeaturesData**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: Supported features of the station.

2.1.10 Station Status Methods

2.1.10.1 SetControlMode

Description: Activates or deactivates the control mode. When the control mode is activated the PC can change or set the station parameters; when deactivated, PC can only monitor or get the station parameters but never set them. It is also important to note that when the control mode is active the station only allow the PC to set its parameters.

Prototype: Public **sub** SetControlMode(ByVal stationID as **Long**,
ByVal mode As **ControlModeConnection)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

mode –ControlModeConnection.**MONITOR** or ControlModeConnection.**CONTROL**.

Returns: Nothing.

2.1.10.2 GetControlMode

Description: Returns the current control mode status. When the control mode is activated the PC can change or set the station parameters, when deactivated PC can only monitor or get the station parameters but never set them. It also is important to note that when the control mode is active the station only allow the PC to set its parameters.

Prototype: Public **function** GetControlMode(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current control mode.

2.1.10.3 GetRemoteMode

Description: Returns the current remote mode status. When the remote mode is activated the PC can change the different station tools status (Sleep, Hibernation, Desolder, Extractor) without the need of manually put the tool on its stand. This method must be called in order to use the methods that change tool's status, which are:

SetPortToolStandStatus

SetPortToolExtractorStatus

SetPortToolDesolderStatus

SetPortToolHeaterStatus

SetPortToolSuctionStatus

It is important to note that when remote mode is active the tool only changes its status by a PC order (using the mentioned methods).

Prototype: Public **function** GetRemoteMode(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current ON or OFF status of the remote mode.

2.1.10.4 SetRemoteMode

Description: Activates or deactivates the remote mode. When the remote mode is activated the PC can change the different station tools status (Sleep, Hibernation, Desolder, Extractor) without the need of manually put the tool on its stand. This method must be called in order to use the methods that change tool's status, which are:

SetPortToolStandStatus

SetPortToolExtractorStatus

SetPortToolDesolderStatus

SetPortToolHeaterStatus

SetPortToolSuctionStatus

It is important to note that when remote mode is active the tool only changes its status by a PC order (using the mentioned methods).

Prototype: Public **sub** SetRemoteMode(ByVal stationID as **Long**, ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

OnOff – The desired ON or OFF status for the remote mode.

Returns: Nothing.

2.1.10.5 GetStationError

Description: Returns the current station error code of the indicated station. See the StationError constant description to more knowledge in the station error codes.

Prototype: Public **function** GetStationError(ByVal stationID as **Long)** as **StationError**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The station error code of the station.

2.1.10.6 GetStationTransformerTemp

Description: Returns the current transformer temperature of the indicated station.

Prototype: Public **function** GetStationTransformerTemp(ByVal stationID as **Long)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The transformer temperature of the station.

2.1.11 Station Data Methods

2.1.11.1 GetStationName

Description: Returns the name of the indicated station.

Prototype: Public **function** GetStationName(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The name of the station.

2.1.11.2 SetStationName

Description: Sets the name of the indicated station.

Prototype: Public **sub** SetStationName(ByVal stationID as **Long**, ByVal newName as **String)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newName – The new desired name for the station.

Returns: Nothing.

2.1.11.3 GetStationInternalUID

Description: Returns the unique ID of the indicated station.

Prototype: Public **function** GetStationInternalUID(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The internal UID (unique ID) of the station.

2.1.11.4 GetStationID

Description: Returns the station ID based on the unique ID of the indicated station.

Prototype: Public **function** GetStationID(ByVal stationUID as **String)** as **Long**

Parameters: **stationUID** – The station UID identifier. This identifier is obtained by calling the "GetStationInternalUID" method.

Returns: The station ID, used in all methods.

2.1.11.5 GetStationPIN

Description: Returns the PIN code of the indicated station.

Prototype: Public **function** GetStationPIN(ByVal stationID as **Long)** as **String**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The PIN code of the station.

2.1.11.6 SetStationPIN

Description: Sets the PIN code of the indicated station.

Prototype: Public **sub** SetStationPIN(ByVal stationID as **Long**, ByVal newPIN as **String)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newPIN – The new desired PIN for the station. A string of 4 numbers.

Returns: Nothing.

2.1.11.7 GetStationPINEnabled

Description: Returns PIN status: PIN is enabled or disabled in the indicated station.

Prototype: Public **function** GetStationPINEnabled(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The status of then PIN.

2.1.11.8 SetStationPINEnabled

Description: Activates or deactivates the PIN in the indicated station.

Prototype: Public **sub** SetStationPINEnabled(ByVal stationID as **Long**, ByVal onoff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

onoff – New status.

Returns: Nothing.

2.1.11.9 GetStationMaxTemp

Description: Returns the maximum temperature of the indicated station.

Prototype: Public **function** GetStationMaxTemp(ByVal stationID as **Long)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The maximum temperature of the station.

2.1.11.10 SetStationMaxTemp

Description: Sets the maximum temperature of the indicated station.

Prototype: Public **sub** SetStationMaxTemp(ByVal stationID as **Long**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newTemp – The desired new maximum temperature.

Returns: Nothing.

2.1.11.11 GetStationMinTemp

Description: Returns the minimum temperature of the indicated station.

Prototype: Public **function** GetStationMinTemp(ByVal stationID as **Long)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The minimum temperature of the station.

2.1.11.12 SetStationMinTemp

Description: Sets the minimum temperature of the indicated station.

Prototype: Public **sub** SetStationMinTemp(ByVal stationID as **Long**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newTemp – The desired new minimum temperature.

Returns: Nothing.

2.1.11.13 GetStationMaxExternalTemp

Description: Returns the maximum external TC temperature of the indicated station.

Prototype: Public **function** GetStationMaxExternalTemp(ByVal stationID as **Long)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The maximum external TC temperature of the station.

2.1.11.14 SetStationMaxExternalTemp

Description: Sets the maximum external TC temperature of the indicated station.

Prototype: Public **sub** SetStationMaxExternalTemp(ByVal stationID as **Long**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newTemp – The desired new maximum external TC temperature.

Returns: Nothing.

2.1.11.15 GetStationMinExternalTemp

Description: Returns the minimum external TC temperature of the indicated station.

Prototype: Public **function** GetStationMinExternalTemp(ByVal stationID as **Long)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The minimum external TC temperature of the station.

2.1.11.16 SetStationMinExternalTemp

Description: Sets the minimum external TC temperature of the indicated station.

Prototype: Public **sub** SetStationMinExternalTemp(ByVal stationID as **Long**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newTemp – The desired new minimum external TC temperature.

Returns: Nothing.

2.1.11.17 GetStationMaxFlow

Description: Returns the maximum flow of the indicated station, in per thousand units.

Prototype: Public **function** GetStationMaxFlow(ByVal stationID as **Long)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The maximum flow of the station, in per thousand units.

2.1.11.18 SetStationMaxFlow

Description: Sets the maximum flow of the indicated station, in per thousand units.

Prototype: Public **sub** SetStationMaxFlow(ByVal stationID as **Long**,
ByVal newFlow as **Integer**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newFlow – The desired new maximum flow, in per thousand units.

Returns: Nothing.

2.1.11.19 GetStationMinFlow

Description: Returns the minimum flow of the indicated station, in per thousand units.

Prototype: Public **function** GetStationMinFlow(ByVal stationID as **Long)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The minimum flow of the station, in per thousand units.

2.1.11.20 SetStationMinFlow

Description: Sets the minimum flow of the indicated station, in per thousand units.

Prototype: Public **sub** SetStationMinFlow(ByVal stationID as **Long**,
ByVal newFlow as **Integer**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newFlow – The desired new minimum flow, in per thousand units.

Returns: Nothing.

2.1.11.21 GetStationTempUnits

Description: Returns the current temperature units used by the indicated station.

Prototype: Public **function** GetStationTempUnits(ByVal stationID as **Long)** as
CTemperature.cTemperatureUnit

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current temperature units used by the station.

2.1.11.22 SetStationTempUnits

Description: Sets the temperature units used by the indicated station.

Prototype: Public **sub** SetStationTempUnits(ByVal stationID as **Long**,
ByVal newUnits as **CTemperature.TemperatureUnit**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newUnits – The desired new temperature units.

Returns: Nothing.

2.1.11.23 GetStationN2Mode

Description: Returns the current nitrogen mode ON or OFF status of the indicated station.

Prototype: Public **function** GetStationN2Mode(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current nitrogen mode ON or OFF status of the station.

2.1.11.24 SetStationN2Mode

Description: Sets the nitrogen mode ON or OFF status of the indicated station.

Prototype: Public **sub** SetStationN2Mode(ByVal stationID as **Long**, ByVal newMode as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newMode – The desired new nitrogen mode status.

Returns: Nothing.

2.1.11.25 GetStationHelpText

Description: Returns the current help text ON or OFF status of the indicated station.

Prototype: Public **function** GetStationHelpText(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current help text ON or OFF status of the station.

2.1.11.26 SetStationHelpText

Description: Sets the help text ON or OFF status of the indicated station.

Prototype: Public **sub** SetStationHelpText(ByVal stationID as **Long**, ByVal newHelp as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

newHelp – The desired new help text status.

Returns: Nothing.

2.1.11.27 GetStationBeep

Description: Returns the current Beep ON or OFF status of the indicated station.

Prototype: Public **function** GetStationBeep(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current Beep ON or OFF status of the station.

2.1.11.28 SetStationBeep

Description: Sets the Beep ON or OFF status of the indicated station.

Prototype: Public **sub** SetStationBeep(ByVal stationID as **Long**, ByVal beep as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

beep – The desired new beep ON or OFF status.

Returns: Nothing.

2.1.11.29 GetStationLocked

Description: Returns the current Locked status of the indicated station.

Prototype: Public **function** GetStationLocked(ByVal stationID as **Long)** as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current Locked ON or OFF status of the station.

2.1.11.30 SetStationLocked

Description: Sets the Locked status of the indicated station.

Prototype: Public **sub** SetStationLocked(ByVal stationID as **Long**, ByVal locked as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

locked – The desired new locked ON or OFF status.

Returns: Nothing.

2.1.12 Remote Mode and Tool Status Methods

2.1.12.1 GetPortToolStandStatus

Description: Returns the current stand ON or OFF status of the tool at the indicated port.

Prototype: Public **function** GetPortToolStandStatus(ByVal stationID as **Long**,
ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the stand mode.

2.1.12.2 GetPortToolSleepStatus

Description: Returns the current sleep ON or OFF status of the tool at the indicated port.

Prototype: Public **function** GetPortToolSleepStatus(ByVal stationID as **Long**,
ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the sleep mode.

2.1.12.3 GetPortToolHibernationStatus

Description: Returns the current hibernation ON or OFF status of the tool at the indicated port.

Prototype: Public **function** GetPortToolHibernationStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the hibernation mode.

2.1.12.4 GetPortToolExtractorStatus

Description: Returns the current extractor ON or OFF status of the tool at the indicated port.

Prototype: Public **function** GetPortToolExtractorStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the extractor mode.

2.1.12.5 GetPortToolDesolderStatus

Description: Returns the current desolder ON or OFF status of the tool at the indicated port.

Prototype: Public **function** GetPortToolDesolderStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the desolder mode.

2.1.12.6 GetPortToolPedalStatus

Description: Returns the current pedal status at the indicated port.

Prototype: Public **function** GetPortToolPedalStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current ON or OFF status of the pedal.

2.1.12.7 GetPortToolPedalConnectedStatus

Description: Returns if a pedal is connected at the indicated port.

Prototype: Public **function** GetPortToolPedalStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if a pedal is connected.

2.1.12.8 GetPortToolSuctionRequestedStatus

Description: Returns if suction was required at the indicated port.

Prototype: Public **function** GetPortToolSuctionRequestedStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if suction was required.

2.1.12.9 GetPortToolSuctionStatus

Description: Returns if tool is suctioning or not at the indicated port.

Prototype: Public **function** GetPortToolSuctionStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if tool is suctioning.

2.1.12.10 GetPortToolHeaterRequestedStatus

Description: Returns if heating was required at the indicated port.

Prototype: Public **function** GetPortToolHeaterRequestedStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if heating was required.

2.1.12.11 GetPortToolHeaterStatus

Description: Returns if tool is heating or not at the indicated port.

Prototype: Public **function** GetPortToolHeaterStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if tool is heating.

2.1.12.12 GetPortToolCoollingStatus

Description: Returns if tool is cooling or not at the indicated port.

Prototype: Public **function** GetPortToolCoolingStatus(ByVal stationID as **Long**, ByVal port as **Port**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: ON if tool is cooling.

2.1.12.13 GetPortToolTimeToStopStatus

Description: Returns the current time to stop, in seconds, at the indicated port.

Prototype: Public **function** GetPortToolTimeToStopStatus(ByVal stationID as **Long**,
ByVal port as **Port**) as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: Time to stop, in seconds.

2.1.12.14 SetPortToolStandStatus

Description: Activates or deactivates stand when the remote mode is ON, see SetRemoteMode(). The stand status is the one you get when the tool is put in the stand.

Prototype: Public **sub** SetPortToolStandStatus(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

OnOff – The desired ON or OFF status for the stand status.

Returns: Nothing.

2.1.12.15 SetPortToolExtractorStatus

Description: Activates or deactivates extractor when the remote mode is ON, see SetRemoteMode().

Prototype: Public **sub** SetPortToolExtractorStatus(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

OnOff – The desired ON or OFF status for the extractor status.

Returns: Nothing.

2.1.12.16 SetPortToolDesolderStatus

Description: Activates or deactivates desolder when the remote mode is ON, see SetRemoteMode().

Prototype: Public **sub** SetPortToolDesolderStatus(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

OnOff – The desired ON or OFF status for the desolder status.

Returns: Nothing.

2.1.12.17 SetPortToolHeaterStatus

Description: Activates or deactivates heater when the remote mode is ON, see SetRemoteMode().

Prototype: Public **sub** SetPortToolHeaterStatus(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

OnOff – The desired ON or OFF status for the heater status.

Returns: Nothing.

2.1.12.18 SetPortToolSuctionStatus

Description: Activates or deactivates suction when the remote mode is ON, see SetRemoteMode().

Prototype: Public **sub** SetPortToolSuctionStatus(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal OnOff as **OnOff)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

OnOff – The desired ON or OFF status for the suction status.

Returns: Nothing.

2.1.13 Port and Tool Info Methods

2.1.13.1 GetPortToolID

Description: Returns the model of the tool at the indicated port.

Prototype: Public **function** GetPortToolID(ByVal stationID as **Long**,
ByVal port as **Port**) as **GenericStationTools**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The model of the tool at the indicated port.

2.1.13.2 GetPortToolActualTemp

Description: Returns the current tip or air temperature of the tool at the indicated port.

Prototype: Public **function** GetPortToolActualTemp(ByVal stationID as **Long**,
ByVal port as **Port**) as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current tip or air temperature of the tool at the indicated port.

2.1.13.3 GetPortToolActualPower

Description: Returns the current power consumption of the tool at the indicated port in per thousand units.

Prototype: Public **function** GetPortToolActualPower(ByVal stationID as **Long**, ByVal port as **Port**) as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current power consumption of the tool at the indicated port.

2.1.13.4 GetPortToolActualFlow

Description: Returns the current air flow, in per thousand units, of the tool at the indicated port.

Prototype: Public **function** GetPortToolActualFlow(ByVal stationID as **Long**, ByVal port as **Port**) as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current air flow of the tool at the indicated port, in per thousand units.

2.1.13.5 GetPortToolActualExternalTemp

Description: Returns the current external TC air temperature at the indicated port.

Prototype: Public **function** GetPortToolActualExternalTemp(ByVal stationID as **Long**, ByVal port as **Port**) as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current external TC temperature at the indicated port.

2.1.13.6 GetPortToolProtectionTCTemp

Description: Returns the current protection TC temperature at the indicated port.

Prototype: Public **function** GetPortToolProtectionTCTemp(ByVal stationID as **Long**, ByVal port as **Port**) as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current protection TC temperature at the indicated port.

2.1.13.7 GetPortToolError

Description: Returns the current error code of the tool at the indicated port. See ToolError constant definitions for more information on tool error codes.

Prototype: Public **function** GetPortToolError(ByVal stationID as **Long**, ByVal port as **Port**) as **ToolError**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current error code of the tool at the indicated port.

2.1.13.8 GetPortToolMOStemp

Description: Returns the current MOSFET temperature of the indicated port.

Prototype: Public **function** GetPortToolMOStemp(ByVal stationID as **Long**, ByVal port as **Port**) as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current MOSFET temperature of the indicated port.

2.1.13.9 GetPortToolFutureMode

Description: Returns the future mode of the tool at the indicated port. The future mode can be sleep or hibernation. This function indicates the next mode the station will achieve if the delay time is not interrupted.

Prototype: Public **function** GetPortToolFutureMode(ByVal stationID as **Long**, ByVal port as **Port**) as **ToolFutureMode**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The future mode of the tool at the indicated port.

2.1.13.10 GetPortToolTimeToFutureMode

Description: Returns the remaining time to the future mode of the tool at the indicated port in seconds. See GetPortToolFutureMode().

Prototype: Public **function** GetPortToolTimeToFutureMode(ByVal stationID as **Long**, ByVal port as **Port**) as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The remaining time to the future mode of the tool at the indicated port.

2.1.14 Port and Tool Data Methods

2.1.14.1 GetPortToolSelectedTemp

Description: Returns the current selected temperature of the tool at the indicated port.

Prototype: Public **function** GetPortToolSelectedTemp(ByVal stationID as **Long**, ByVal port as **Port)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current selected temperature of the tool at the indicated port.

2.1.14.2 SetPortToolSelectedTemp

Description: Sets the selected temperature of the tool at the indicated port.

Prototype: Public **sub** SetPortToolSelectedTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

newTemp – The desired new selected temperature.

Returns: Nothing.

2.1.14.3 GetPortToolSelectedFlow

Description: Returns the current selected flow at the indicated port, in per thousand units.

Prototype: Public **function** GetPortToolSelectedFlow(ByVal stationID as **Long**,
ByVal port as **Port)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current selected flow at the indicated port, in per thousand units.

2.1.14.4 SetPortToolSelectedFlow

Description: Sets the selected flow at the indicated port, in per thousand units.

Prototype: Public **sub** SetPortToolSelectedFlow(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal newFlow as **Integer**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

newFlow – The desired new selected flow, in per thousand units.

Returns: Nothing.

2.1.14.5 GetPortToolSelectedExternalTemp

Description: Returns the current selected external temperature of the tool at the indicated port.

Prototype: Public **function** GetPortToolSelectedExternalTemp(ByVal stationID as **Long**,
ByVal port as **Port)** as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current selected external temperature of the tool at the indicated port.

2.1.14.6 SetPortToolSelectedExternalTemp

Description: Sets the selected external temperature of the tool at the indicated port.

Prototype: Public **sub** SetPortToolSelectedTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

newTemp – The desired new selected temperature.

Returns: Nothing.

2.1.14.7 GetPortWorkMode

Description: Returns the current work mode at the indicated port.

Prototype: Public **function** GetPortWorkMode(ByVal stationID as **Long**,
ByVal port as **Port)** as **PortWorkMode_HA**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

Returns: The current work mode at the indicated port.

2.1.14.8 SetPortWorkMode

Description: Sets the work mode for the tool at the indicated port.

Prototype: Public **sub** SetPortWorkMode(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal newMode as **PortWorkMode_HA**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

newMode – The desired work mode.

Returns: Nothing.

2.1.14.9 GetPortToolFixTemp

Description: Returns the current selected fix temperature for the tool model at the indicated port.

Prototype: Public **function** GetPortToolFixTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
Ctemperature

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The current selected fix temperature for the tool model at the indicated port.

2.1.14.10 SetPortToolFixTemp

Description: Sets the selected fix temperature for the tool model at the indicated port.

Prototypes: Public **sub** SetPortToolFixTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal newTemp as **Ctemperature**)

Public **sub** SetPortToolFixTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal newOnOff as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

newTemp – The desired new fix temperature.

newOnOff – Use **OnOff._OFF** to deactivate fix temperature.

Returns: Nothing.

2.1.14.11 GetPortToolSelectedTempLevels

Description: Returns the current selected temperature level for the tool model at the indicated port.

Prototype: Public **function** GetPortToolSelectedTempLevels(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**)
as **ToolTemperatureLevels**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The current selected temperature levels for the tool model at the indicated port.

2.1.14.12 GetPortToolSelectedTempLevelsEnabled

Description: Returns if the temperature levels are active, for the tool model at the indicated port.

Prototype: Public **function** GetPortToolSelectedTempLevelsEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**)
as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: If the temperature levels are active or not for the tool model at the indicated port.

2.1.14.13 SetPortToolSelectedTempLevels

Description: Sets the temperature level for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolSelectedTempLevels(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal levels as **ToolTemperatureLevels**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

levels – The desired new temperature levels.

Returns: Nothing.

2.1.14.14 SetPortToolSelectedTempLevelsEnabled

Description: Activates or deactivates temperature levels for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolSelectedTempLevelsEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal onoff as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

onoff – Enable or disable temperature levels.

Returns: Nothing.

2.1.14.15 GetPortToolTempLevel

Description: Returns the current temperature of the indicated level for the tool model at indicated port.

Prototype: Public **function** GetPortToolTempLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**) as
Ctemperature

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired temperature level.

Returns: The current temperature of the indicated level for the tool model at the indicated port.

2.1.14.16 SetPortToolTempLevel

Description: Sets the temperature of the indicated level for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolTempLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired temperature level.

newTemp – The desired new temperature for the level.

Returns: Nothing.

2.1.14.17 GetPortToolFlowLevel

Description: Returns the current flow, in per thousand units, of the indicated level for the tool model at indicated port.

Prototype: Public **function** GetPortToolFlowLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**) as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired level.

Returns: The current flow of the indicated level for the tool model at the indicated port.

2.1.14.18 SetPortToolFlowLevel

Description: Sets the flow, in per thousand units, of the indicated level for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolFlowLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**,
ByVal newFlow as **Integer**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired temperature level.

newFlow – The desired new flow for the level.

Returns: Nothing.

2.1.14.19 GetPortToolExternalTempLevel

Description: Returns the temperature for the external TC of the indicated level for the tool model at indicated port.

Prototype: Public **function** GetPortToolExternalTempLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**) as **Ctemperature**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired external temperature level.

Returns: The external temperature of the indicated level for the tool model at the indicated port.

2.1.14.20 SetPortToolExternalTempLevel

Description: Sets the temperature for the external TC of the indicated level for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolExternalTempLevel(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired external temperature level.

newTemp – The desired new temperature for the level.

Returns: Nothing.

2.1.14.21 GetPortToolTempLevelEnabled

Description: Returns if the level is active for the tool model at indicated port.

Prototype: Public **function** GetPortToolTempLevelEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**) as **OnOff**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired temperature level.

Returns: If the temperature level is active or not for the tool model at the indicated port.

2.1.14.22 SetPortToolTempLevelEnabled

Description: Activates or deactivates the level for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolTempLevelEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal level as **ToolTemperatureLevels**,
ByVal onoff as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

level – The desired temperature level.

onoff – Enable or disable the level.

Returns: Nothing.

2.1.14.23 SetPortToolLevels

Description: Sets the selected level and temperature levels of the indicated port of the indicated station, at once.

Prototype: Public **sub** SetPortToolLevels(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal selectedLevelEnabled as **OnOff**,
ByVal selectedLevel as **ToolTemperatureLevels**
ByVal level1Enabled as **OnOff**,
ByVal tempLevel1 as **Ctemperatur**
ByVal level2Enabled as **OnOff**,
ByVal tempLevel2 as **Ctemperature**
ByVal level3Enabled as **OnOff**,
ByVal tempLevel3 as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

selectedLevelEnabled – Activates or deactivates temperature levels.

selectedLevel – The select the temperature level.

levelxEnabled – Enable or disable the level.

tempLevelx – Level temperature.

Returns: Nothing.

2.1.14.24 SetPortToolLevels_HA

Description: Sets the selected level and temperature levels of the indicated port of the indicated station, at once, for hot air desoldering stations.

Prototype: Public **sub** SetPortToolLevels_HA(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal selectedLevelEnabled as **OnOff**,
ByVal selectedLevel as **ToolTemperatureLevels**
ByVal level1Enabled as **OnOff**,
ByVal tempLevel1 as **Ctemperatur**
ByVal flowLevel1 as **Integer**
ByVal tempExternalLevel1 as **Ctemperatur**
ByVal level2Enabled as **OnOff**,
ByVal tempLevel2 as **Ctemperature**
ByVal flowLevel2 as **Integer**
ByVal tempExternalLevel2 as **Ctemperatur**
ByVal level3Enabled as **OnOff**,
ByVal tempLevel3 as **Ctemperature**
ByVal flowLevel3 as **Integer**
ByVal tempExternalLevel3 as **Ctemperatur**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

selectedLevelEnabled – Activates or deactivates temperature levels.

selectedLevel – The select the temperature level.

levelxEnabled – Enable or disable the level.

tempLevelx – Level temperature.

flowLevelx – Level flow, in per thousand units.

tempExternalLevelx – Level temperature for external TC.

Returns: Nothing.

2.1.14.25 GetPortToolSleepDelay

Description: Returns the sleep delay time defined for the tool model at the indicated port.

Prototype: Public **function** GetPortToolSleepDelay(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
ToolTimeSleep

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The sleep delay time defined for the tool model at the indicated port.

2.1.14.26 GetPortToolSleepDelayEnabled

Description: Returns if the sleep delay time is active for the tool model at the indicated port.

Prototype: Public **function** GetPortToolSleepDelayEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
OnOff

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: If sleep delay time is active or not for the tool model at the indicated port.

2.1.14.27 SetPortToolSleepDelay

Description: Sets the sleep delay time for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolSleepDelay(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal delay as **ToolTimeSleep**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

delay – The desired sleep delay time.

Returns: Nothing.

2.1.14.28 SetPortToolSleepDelayEnabled

Description: Activates or deactivates the sleep delay for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolSleepDelayEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal onoff as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

onoff – Enable or disable sleep delay.

Returns: Nothing.

2.1.14.29 GetPortToolSleepTemp

Description: Returns the current sleep temperature for the tool model at the indicated port.

Prototype: Public **function** GetPortToolSleepTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
Ctemperature

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The current sleep temperature for the tool model at the indicated port.

2.1.14.30 SetPortToolSleepTemp

Description: Sets the sleep temperature for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolSleepTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

newTemp – The desired sleep temperature.

Returns: Nothing.

2.1.14.31 GetPortToolHibernationDelay

Description: Returns the current hibernation delay time for the tool model at the indicated port.

Prototype: Public **function** GetPortToolHibernationDelay(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
ToolTimeHibernation

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The current hibernation delay time for the tool model at the indicated port.

2.1.14.32 GetPortToolHibernationDelayEnabled

Description: Returns if the hibernation delay is active for the tool model at the indicated port.

Prototype: Public **function** GetPortToolHibernationDelayEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
OnOff

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: If the hibernation delay is active or not for the tool model at the indicated port.

2.1.14.33 SetPortToolHibernationDelay

Description: Sets the hibernation delay time for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolHibernationDelay(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal delay as **ToolTimeHibernation**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

delay – The desired hibernation delay time.

Returns: Nothing.

2.1.14.34 SetPortToolHibernationDelayEnabled

Description: Activates or deactivates the hibernation delay for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolHibernationDelayEnabled(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal onoff as **OnOff**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

onoff – Enable or disable the hibernation delay.

Returns: Nothing.

2.1.14.35 GetPortToolAdjustTemp

Description: Returns the current adjust temperature for the tool model at the indicated port.

Prototype: Public **function** GetPortToolAdjustTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
Ctemperature

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The current adjust temperature for the tool model at the indicated port.

2.1.14.36 SetPortToolAdjustTemp

Description: Sets the adjust temperature for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolAdjustTemp(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal newTemp as **Ctemperature**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

newTemp – The desired adjust temperature.

Returns: Nothing.

2.1.14.37 GetPortToolTimeToStop

Description: Returns the time to stop in seconds defined for the tool model at the indicated port.

Prototype: Public **function** GetPortToolTimeToStop(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
Integer

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The time to stop defined for the tool model at the indicated port.

2.1.14.38 SetPortToolTimeToStop

Description: Sets the time to stop defined for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolTimeToStop(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal newValue as **Integer**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

newValue – The desired time to stop in seconds.

Returns: Nothing.

2.1.14.39 GetPortToolStartMode

Description: Returns the start mode defined for the tool model at the indicated port.

Prototype: Public **function** GetPortToolTimeToStop(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**) as
CToolStartMode_HA

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: The CToolStartMode_HA structure defined for the tool model at the indicated port.

2.1.14.40 SetPortToolStartMode

Description: Sets the start mode defined for the tool model at the indicated port.

Prototype: Public **sub** SetPortToolStartMode(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**,
ByVal mode as **CToolStartMode_HA**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

mode – The desired start mode structure.

Returns: Nothing.

2.1.14.41 ResetPortToolSettings

Description: *Reset data* for the tool model at the indicated port.

Prototype: Public **sub** ResetPortToolSettings(ByVal stationID as **Long**,
ByVal port as **Port**,
ByVal tool as **GenericStationTools**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port where the desired tool is.

tool – The desired tool model.

Returns: Nothing.

2.1.15 Counters methods

2.1.15.1 GetStationPluggedMinutes

Description: Returns the current number of connected minutes of the indicated station.

Prototype: Public **function** GetStationPluggedMinutes(ByVal stationID as **Long,** Optional
ByVal counterType As **CounterTypes)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The number of connected minutes of the station.

2.1.15.2 GetPortToolWorkMinutes

Description: Returns the number of working minutes of the tool at the indicated port.

Prototype: Public **function** GetPortToolWorkMinutes(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The working minutes of the tool at the indicated port.

2.1.15.3 GetPortToolSleepMinutes

Description: Returns the number of sleep minutes of the tool at the indicated port.

Prototype: Public **function** GetPortToolSleepMinutes(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The sleep minutes of the tool at the indicated port.

2.1.15.4 GetPortToolHibernationMinutes

Description: Returns the number of hibernation minutes of the tool at the indicated port.

Prototype: Public **function** GetPortToolHibernationMinutes(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The hibernation minutes of the tool at the indicated port.

2.1.15.5 GetPortToolIdleMinutes

Description: Returns the number of idle minutes of the tool at the indicated port. Idle means no tool connected at the port.

Prototype: Public **function** GetPortToolIdleMinutes(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The idle minutes of the tool at the indicated port.

2.1.15.6 GetPortToolSleepCycles

Description: Returns the number of sleep cycles of the tool at the indicated port.

Prototype: Public **function** GetPortToolSleepCycles(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The sleep cycles of the tool at the indicated port.

2.1.15.7 GetPortToolDesolderCycles

Description: Returns the number of desolder cycles of the tool at the indicated port.

Prototype: Public **function** GetPortToolDesolderCycles(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The desolder cycles of the tool at the indicated port.

2.1.15.8 GetPortToolWorkCycles

Description: Returns the number of working cycles of the tool at the indicated port.

Prototype: Public **function** GetPortToolWorkCycles(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The working cycles of the tool at the indicated port.

2.1.15.9 GetPortToolSuctionCycles

Description: Returns the number of suction cycles of the tool at the indicated port.

Prototype: Public **function** GetPortToolSuctionCycles(ByVal stationID as **Long**,
ByVal port as **Port,**
Optional ByVal counterType As **CounterTypes**)
as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

counterType - GLOBAL_COUNTER (default) or PARTIAL_COUNTER

Returns: The suction cycles of the tool at the indicated port.

2.1.15.10 ResetPortToolStationPartialCounters

Description: Resets all partial counters of the indicated port of the indicated station.

Prototype: Public **function** ResetPortToolStationPartialCounter(ByVal stationID as **Long**,
ByVal port as **Port**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

port – The identifier of the port.

Returns: Nothing.

2.1.16 Continuous Mode Methods

2.1.16.1 SetContinuousMode

Description: Defines the speed and ports to be used in continues mode. When the continuous mode is active the station sends port data automatically at the indicated speed (actually it's a period). The data sent is internally stored in one or more FIFO queues that can be accessed using GetContinuousModeNextData(). The maximum length of data that the queue stores is indicated in the constant:

CONTINUOUS_MODE_QUEUE_MAX_LENGTH

The port data is composed basically of the current port tool tip temperature and power consumption, it also contains a sequence number used to know the order of the transmissions and possible data transmission lost. For Hot Air desoldering stations flow and external temperatures are added.

The desired ports to activate can be from 1 to 4 (depending on the station model) and must be indicated to the function.

Prototype: Public **sub** SetContinuousMode(ByVal stationID as **Long**,
ByVal speed as **SpeedContinuousMode**,
Optional ByVal portA as **Port** = Port.NO_PORT,
Optional ByVal portB as **Port** = Port.NO_PORT,
Optional ByVal portC as **Port** = Port.NO_PORT,
Optional ByVal portD as **Port** = Port.NO_PORT**)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

speed – The desired speed (period) value for the continuous mode.

portA – The first desired port where to activate the continuous mode.

portB – The second desired port where to activate the continuous mode.

portC – The third desired port where to activate the continuous mode.

portD – The fourth desired port where to activate the continuous mode.

Returns: Nothing.

2.1.16.2 StartContinuousMode

Description: Starts a new continuous data queue instance for this station. If it is the first queue instance for the station, the station continuous mode is started. It returns a queue ID to be used by GetContinuousModeNextData.

The station transmision speed and ports will be the ones defined in SetContinuousMode method, but the delivery speed may be diferent for each queue instance.

Prototype: Public **function** StartContinuousMode(ByVal stationID as **Long**, optional deliverSpeed as **SpeedContinuousMode**) as **UInteger**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: A queue ID.

2.1.16.3 GetContinuousModeDeliverySpeed

Description: Retrieves the speed defined for the queue in StartContinuousMode

The station transmision speed and ports will be the ones defined in SetContinuousMode method, but the delivery speed may be diferent for each queue instance.

Prototype: Public **function** GetContinuousModeDeliverySpeed(ByVal stationID as **Long**, ByVal queueID as **UInteger**) as **SpeedContinuousMode**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

queueID – The identifier of the queue.

Returns: The delivery speed.

2.1.16.4 StopContinuousMode

Description: Stops a continuous data queue instance. If there are no more active queues for this station, continues mode in the station is stopped.

Prototype: Public **sub** StopContinuousMode(ByVal stationID as **Long**, ByVal queueID as **UInteger)**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Parameters: **queueID** – The identifier of queue instance to stop.

Returns: Nothing.

2.1.16.5 GetContinuousMode

Description: Returns the current continuous mode status. Indicates the current speed (period) and the active ports in a special structure.

Prototype: Public **function** GetContinuousMode(ByVal stationID as **Long)** as **CContinuousModeStatus**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: The current continuous mode status of the station.

2.1.16.6 GetContinuousModeDataCount

Description: Returns the current continuous mode pending data in the internal queue. The data in the queue is extracted by calling GetContinuousModeNextData().

Prototype: Public **function** GetContinuousModeDataCount(ByVal stationID as **Long**, ByVal queueID as **UInteger)** as **Integer**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Parameters: **queueID** – The identifier of queue instance.

Returns: The current continuous mode internal queue pending data to extract.

2.1.16.7 GetContinuousModeNextData

Description: Returns the next data in the continuous mode queue and removes it from the queue.

Prototype: Public **function** GetContinuousModeNextData(ByVal stationID as **Long,** ByVal queueID as **UInteger)** as **stContinuousModeData**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Parameters: **queueID** – The identifier of queue instance.

Returns: The next data in the continuous mode queue.

2.1.16.8 GetContinuousModeNextData_HA

Description: Returns the next data in the continuous mode queue and removes it from the queue.

Prototype: Public **function** GetContinuousModeNextData_HA(ByVal stationID as **Long,** ByVal queueID as **UInteger)** as **stContinuousModeData_HA**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Parameters: **queueID** – The identifier of queue instance.

Returns: The next data in the continuous mode queue.

2.1.17 Miscelaneous Methods

2.1.17.1 DefaultStationParameters

Description: Restores default values for all the parameters of the indicated station.

Prototype: Public **sub** DefaultStationParameters(ByVal stationID as **Long**)

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: Nothing.

2.1.17.2 SetTransaction

Description: The library will raise a TransactionFinished(transactionID) event when receive the response from the station. You can use this function to know that all previous functions have been processed by the station.

Prototype: Public **function** SetTransaction(ByVal stationID as **Long)** as **UInteger**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

Returns: A transaction number that will be received within the *TransactionFinished event*.

2.1.17.3 QueryTransaction

Description: You can use this function to know if a transaction ID has finished, instead of the *TransactionFinished event.*

Prototype: Public **function** QueryTransaction(ByVal stationID as **Long**,
ByVal transactionID as **UInteger)** as **boolean**

Parameters: **stationID** – The identifier of the desired station, this identifier is obtained by getting the "NewStationConnected" event.

transactionID – The transaction ID returned in SetTransaction function.

Returns: True if transaction ID has finished.

2.2 Ctemperature Class

This is a tool class used to represent temperatures. Some of the API methods use this class in order to pass and receive temperature parameters.

This class stores the temperature in UTI units and is designed to easily set and get the temperature in Fahrenheit and in Celsius degrees.

Following are the methods and properties of this class:

1. **UTI**:

   Description: Stores and retrieves the temperature value in UTI.

   Prototype: Public **Property** UTI as **Integer**

2. **ToRoundCelsius**

   Description: Returns the temperature in Celsius degrees in steps of five.

   Prototype: Public **Function** ToRoundCelsius() as **Integer**

3. **ToCelsius**

   Description: Returns the temperature in Celsius degrees.

   Prototype: Public **Function** ToCelsius() as **Integer**

4. **ToRoundFahrenheit**

   Description: Returns the temperature in Fahrenheit degrees in steps of ten.

   Prototype: Public **Function** ToRoundFahrenheit() as **Integer**

5. **ToFahrenheit**

   Description: Returns the temperature in Fahrenheit degrees.

   Prototype: Public **Function** ToFahrenheit() as **Integer**

6. **InCelsius**

   Description: Sets the temperature in Celsius degrees.

   Prototype: Public **Sub** InCelsius(ByVal value as **Integer**)

   Parameters: **value** – The desired temperature in Celsius

7. **InFahrenheit**

   Description: Sets the temperature in Fahrenheit degrees.

   Prototype: Public **Sub** InFahrenheit(ByVal value as **Integer**)

   Parameters: **value** – The desired temperature in Fahrenheit

8. **ToCelsiusToAdjust**

   Description: Returns the temperature in Celsius degrees of an adjust temperature.

   Prototype: Public **Function** ToCelsiusToAjust() as **Integer**

9. **ToFahrenheitToAdjust**

   Description: Returns the temperature in Fahrenheit degrees of an adjust temperature.

   Prototype: Public **Function** ToFahrenheitToAdjust() as **Integer**

10. **InCelsiusToAdjust**

    Description: Sets the temperature in Celsius degrees of an adjust temperature.

    Prototype: Public **Sub** InCelsiusToAdjust(ByVal value as **Integer**)

    Parameters: **value** – The desired temperature in Celsius

11. **InFahrenheitToAdjust**

    Description: Sets the temperature in Fahrenheit degrees of an adjust temperature.

    Prototype: Public **Sub** InFahrenheitToAdjust(ByVal value as **Integer**)

    Parameters: **value** – The desired temperature in Fahrenheit

12. **isValid**

    Description: Indicates if the temperature value is in range. A temperature that is not valid should not be accepted as the temperature of the corresponding parameter.

    Prototype: Public **Function** isValid() as **Boolean**

2.3 Cerror Class

This is the class for the user errors. User errors refer to bad parameter values passed to the API methods. When these methods detect an error throw the "UserError" event and pass the error as an object of this class.

Following are the methods of this class:

1. **cErrorCodes**:

   Description: List of constants that are the possible user error codes.

   ```
   STATION_ID_NOT_FOUND
   CONTINUOUS_MODE_ON_WITHOUT_PORTS
   PORT_NOT_IN_RANGE
   INVALID_STATION_NAME
   INVALID_STATION_PIN
   TEMPERATURE_OUT_OF_RANGE
   STATION_ID_OVERFLOW
   POWER_LIMIT_OUT_OF_RANGE
   TOOL_NOT_SUPPORTED
   FUNCTION_NOT_SUPPORTED
   COMMUNICATION_ERROR
   PERIPHERAL_NOT_IN_RANGE
   FLOW_OUT_OF_RANGE
   ```

2. **GetMsg**

   Description: Returns the user error message.

   Prototype: Public **Function** GetMsg() as **String**

3. **GetCode**

   Description: Returns the user error code.

   Prototype: Public **Function** GetCode() as **cErrorCodes**

4. **show**

   Description: Shows a MsgBox with error code and the error message.

   Prototype: Public **Sub** show**()**

2.4 History

2.4.1 Version 3.19.01.7

- Hot Air desoldering stations suppport

2.4.2 Version 3.15.01.4

- Partial counters
- Multiple queues for continuous mode
- QueryTransaction function

2.4.3 Version 2.14.09.0

Ethernet support

2.4.4 Version 1.66.76.0

- Added the following methods to Ctemperature class:
  - ToCelsiusToAdjust
  - ToFahrenheitToAdjust
  - InCelsiusToAdjust
  - InFahrenheitToAdjust
- Removed GetStationPowerLimit and SetStationPowerLimit functions.

2.4.5 Version 1.66.75.1

- Added SetTransaction function and TransactionFinished event

2.4.6 Version 1.66.74.1

- Added GetPortToolStandStatus function
- Changed DefaultStationParametters to DefaultStationParameters
- Added optional paramater when creating the API object: Stations Initial Control Mode

-

2.5 Installing USB driver for the new Excellence series stations

The first time you connect a station from the Excellence series to a PC, you must install the required USB driver.

2.5.1 For Windows XP

1. When connecting the USB, the new hardware wizard window appears. Answer "**No, not this time**" to the question about connecting to Windows Update, and press Next.
2. Select the second option: **Install from a list or specific location (Advanced)**, and press Next.
3. Select **Don't search. I will choose the driver to install** and press Next.
4. Click the **Have Disk...** button on the *Select the device driver you want to install for this hardware.*
5. Click the **Browse...** button on the *Install From Disk* dialog box that appeared
6. In the *Locate File* dialog box that appears next, navigate to the folder containing the driver (drivers\Freescale)
7. Click the **INF** file that displays in the file list and click the **Open** button
8. Click the **OK** button back on the *Install From Disk* dialog box.
9. Choose the newly added hardware back on the *Select the device driver you want to install for this hardware.* window and then click the **Next >** button.
10. If you're warned about the software for the hardware device not passing the Windows Logo testing, click the **Continue** button.
11. At the last window, check if the driver was successfully installed, and press **Finish**.

2.5.2 For Windows 7/8

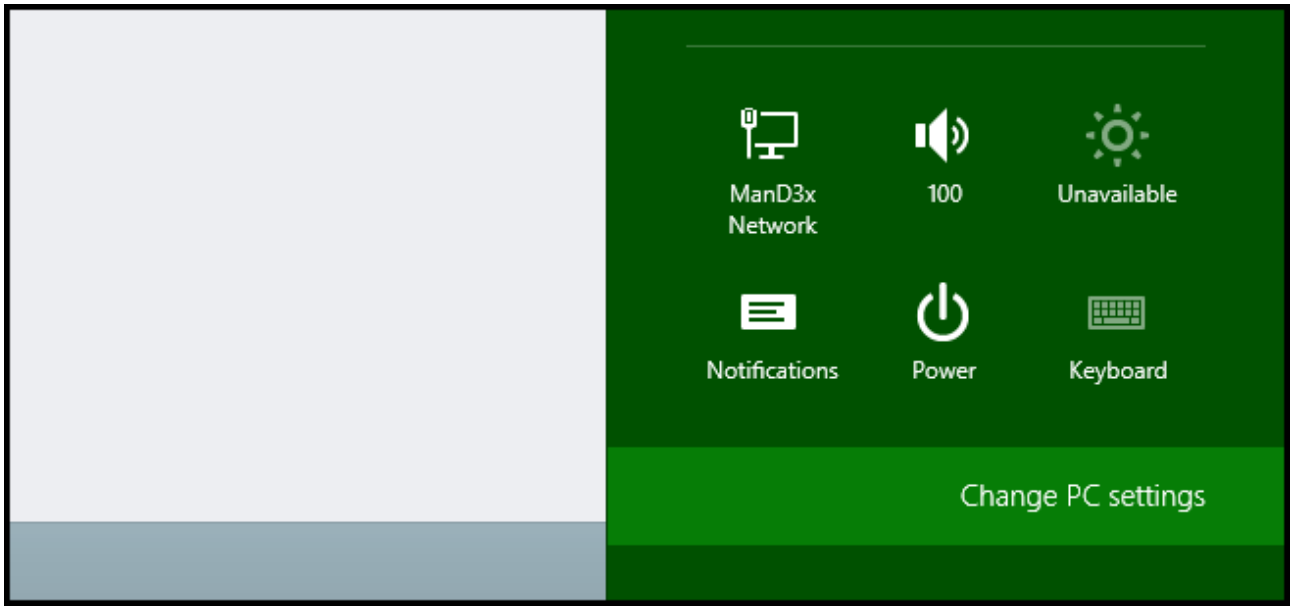
1. When connecting the USB, a message appears stating that the system cannot install the driver.
2. Go to the **Start** icon and select **Control Panel**
3. Search for and select **Device Manager** (or Hardware and Sound and then Device Manager, if view Small/Large icons is selected)
4. Search for **MQX VIRTUAL COM** device.
5. Right click on it and select **Properties**
6. Click the **Driver tab** and press **Update Driver**
7. On the "How do you want to search for driver software?" window, click on **Browse my computer for driver software**.
8. In the next window labeled "Browse for driver software on your computer", click on **Let me pick from a list of device drivers on my computer**, located at the bottom of the window.
9. Click the **Have Disk...** button located under the text box.
10. Click the **Browse...** button on the Install From Disk dialog box that appeared.
    In the Locate File dialog box that appears next, navigate to the folder containing the driver (drivers\Freescale)
11. Click the **INF** file that displays in the file list and click the **Open** button.

2.5.3 If USB driver installation failed on Windows 8

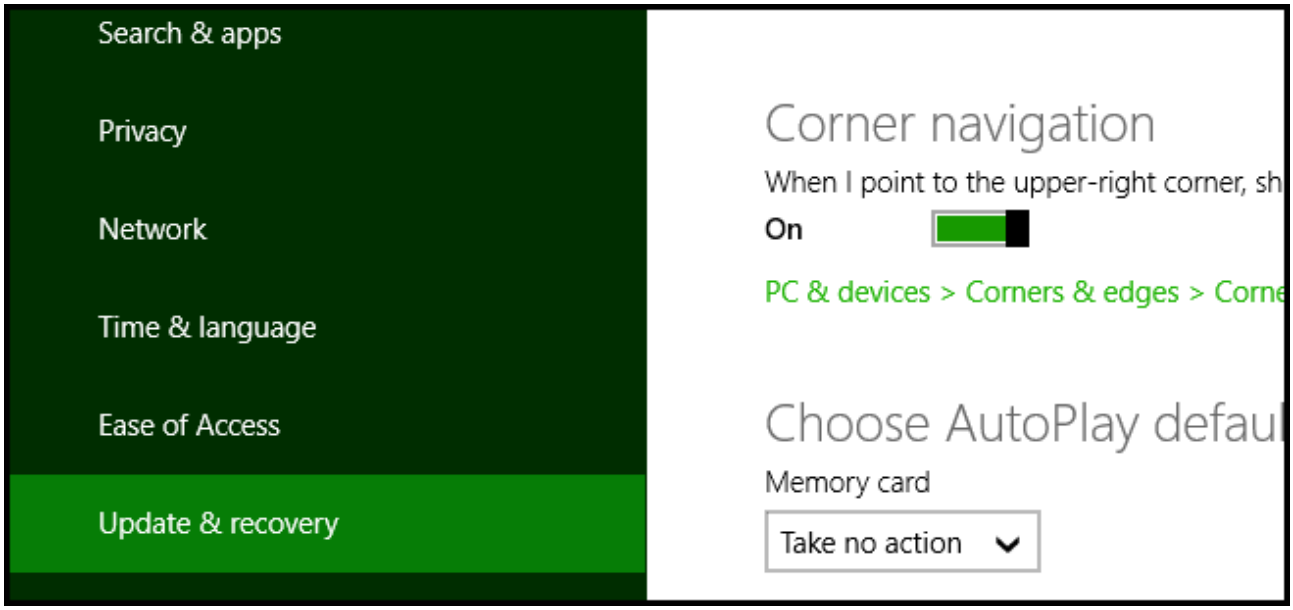
May be Windows 8 blocks the installation of digitally unsigned drivers. To disable driver signature verification on Windows 8, do the following procedure and try it again.

Press the Win + C keyboard combination to bring up the Charms Bar, then click on the Settings Charm.

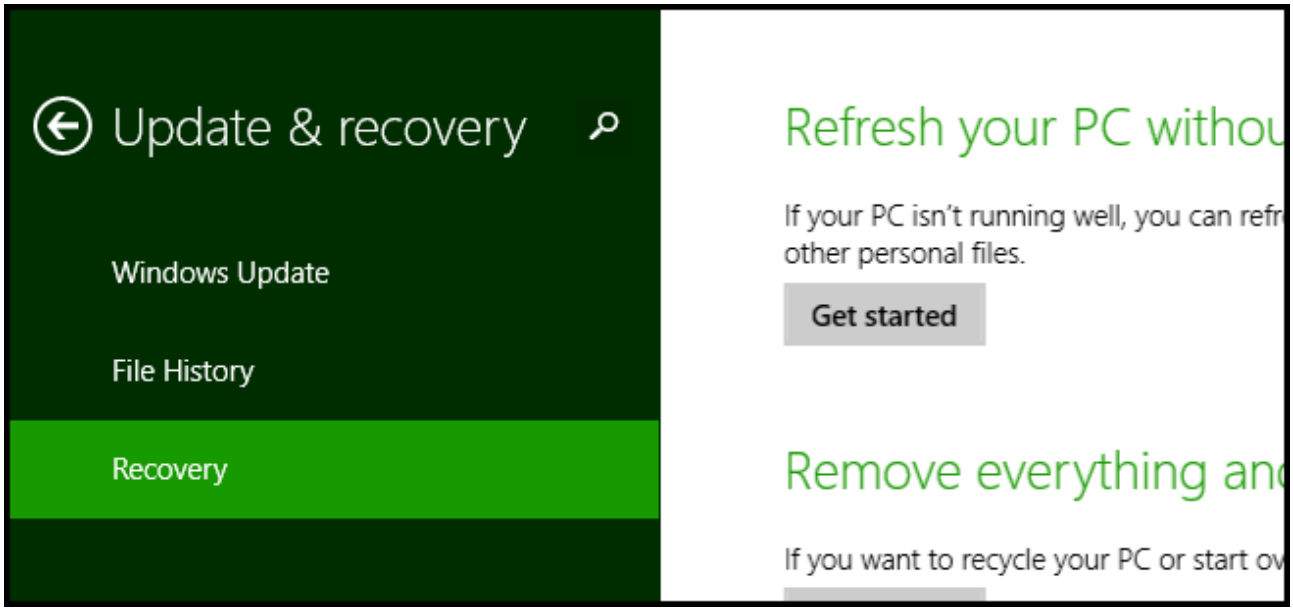
We need to head into the Modern Control Panel, so go ahead and click on the Change PC settings link.



When the Control Panel opens, switch over to the “Update & recovery” section.

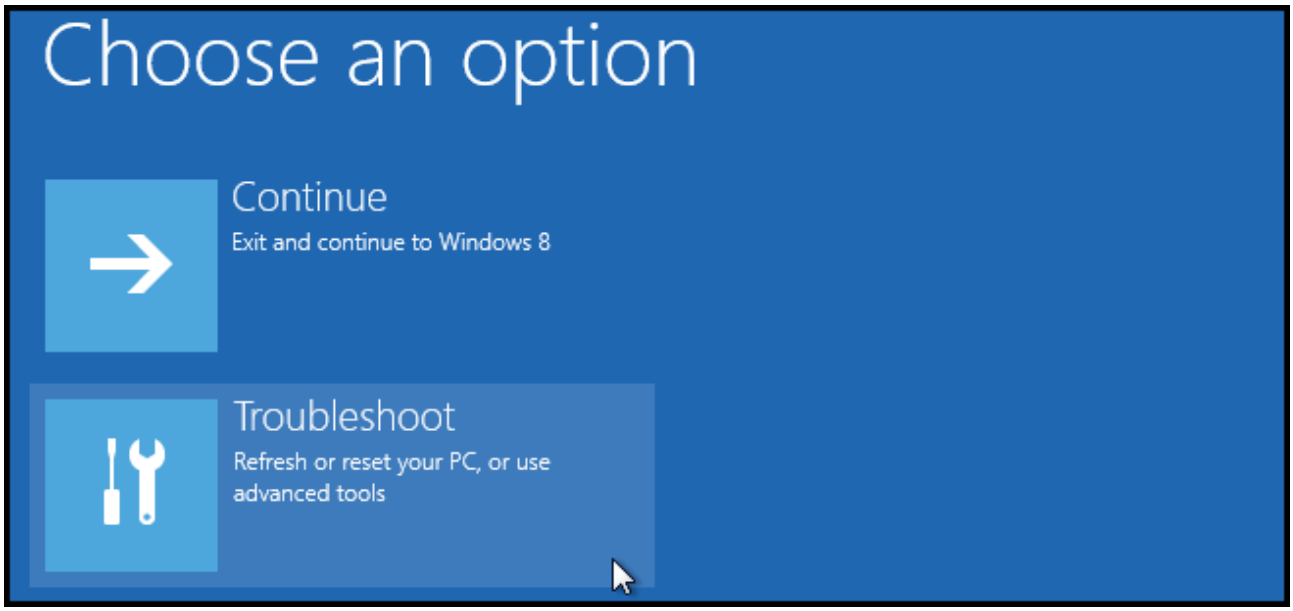


Then click on the Recovery option on the left hand side.

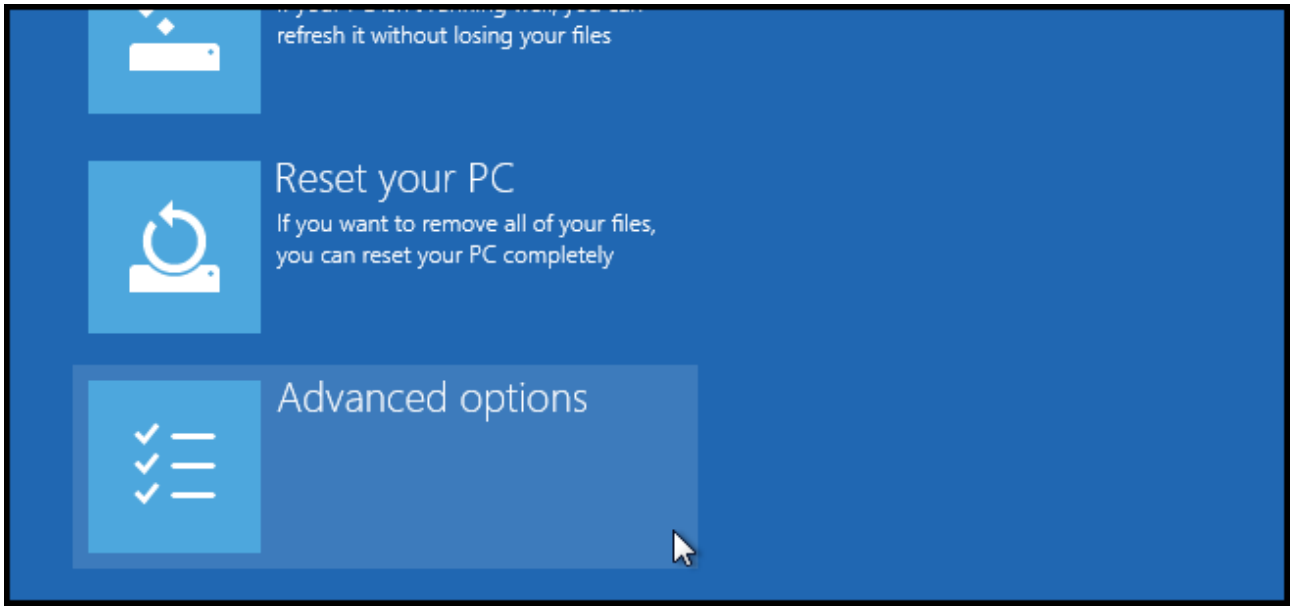


Once selected, you will see an advanced startup section appear on the right hand side. You will need to click on the "Restart now" button.

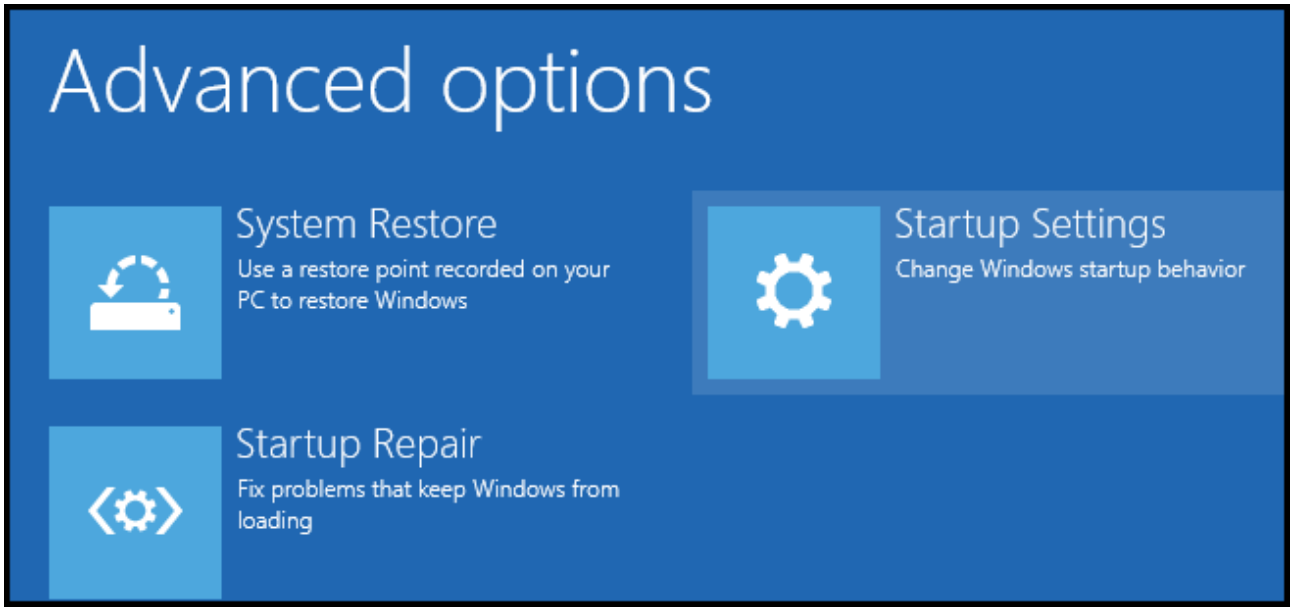
Once your Computer has rebooted you will need to choose the Troubleshoot option.



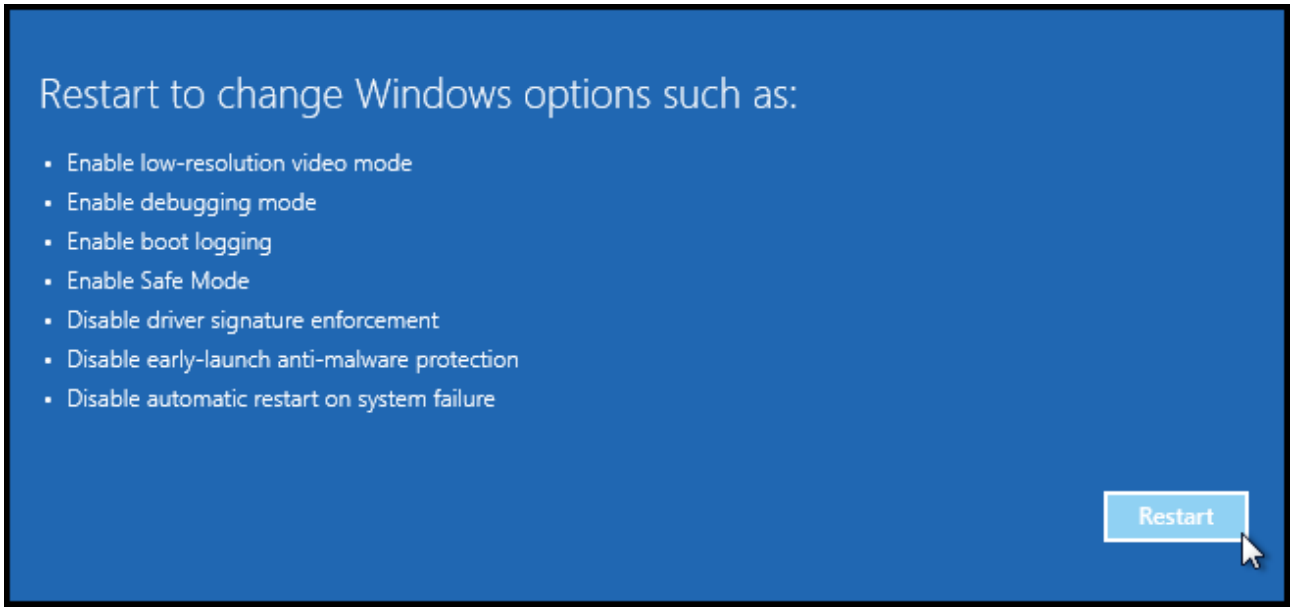
Then head into Advanced options.



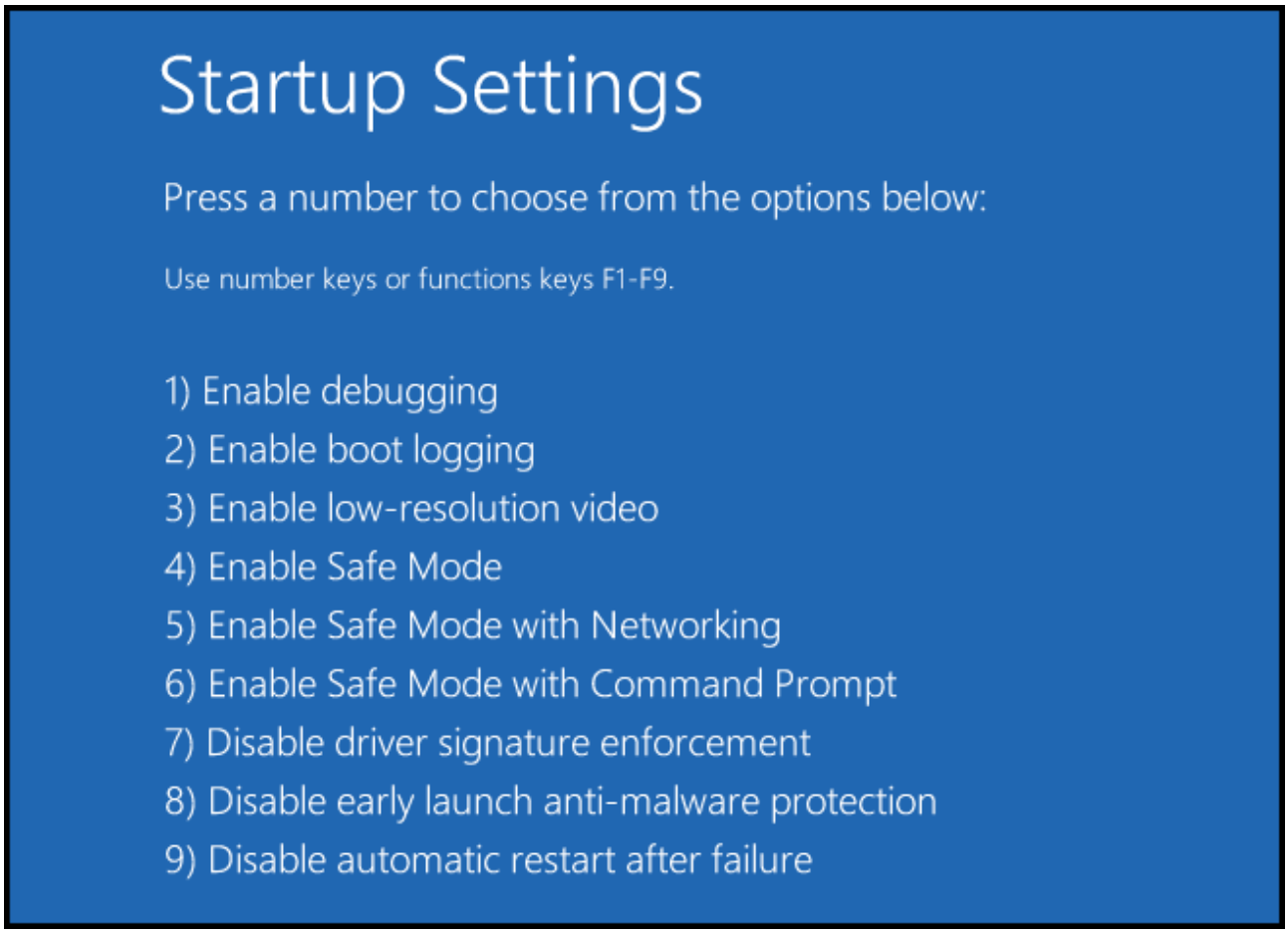
Then Startup Settings.



Since we are modifying boot time configuration settings, you will need to restart your Computer one last time.



Finally, you will be given a list of startup settings that you can change. The one we are looking for is "Disable driver signature enforcement". To choose the setting, you will need to press the F7 key.



That's all. Your PC will then reboot and you will be able to install unsigned drivers

MORE INFORMATION
www.jbctools.com