EC/CC200

e-factory.Basecom
Funktionsbeschreibung V 1.00

Originalbetriebsanleitung

ENGEL
be the first.

Änderungsverzeichnis

Versionsänd. von / auf	Datum	geänderte Seiten	Beschreibung	geändert von
V1.00	20.10.2010		neu erstellt / abgeleitet von Kemro.opc 1.8	sg

Inhalt

1 Einleitung

Der **E-factory.Basecom** ist ein OPC Server. Durch die Realisierung des **E-factory.Basecom** wird der standardisierte und einfache Zugang zu Daten und Funktionalitäten der Steuerung EC/CC200 möglich.

Da die Schnittstelle des **E-factory.Basecom** dem OPC Standard "OPC Data Access Custom Interface Version 2.04" entspricht, kann jedes Programm (jeder OPC Client), das diesen Standard unterstützt, mit einer EC/CC200 Steuerung kommunizieren.

Um mit diversen Automatisierungsprogrammen über Visual Basic auf die OPC-Daten zuzugreifen, ist der Einsatz einer "Wrapper-DLL" notwendig.

Die notwendigen Schnittstellen sind im OPC Standard "Data Access Automation Interface Standard" definiert.
Allgemeine OPC-Informationen siehe: www.opcfoundation.org

2 Überblick

Der **E-factory.Basecom** läuft unter dem Betriebssystem Windows 2000, WindowsXP und Windows Vista. Er ermöglicht es beliebig vielen OPC Clients, sich zu beliebig vielen (nur eingeschränkt durch die Leistungsfähigkeit des Netzwerkes) EC/CC200 Steuerungen zu verbinden und mit diesen zu kommunizieren (siehe folgende Abbildung)

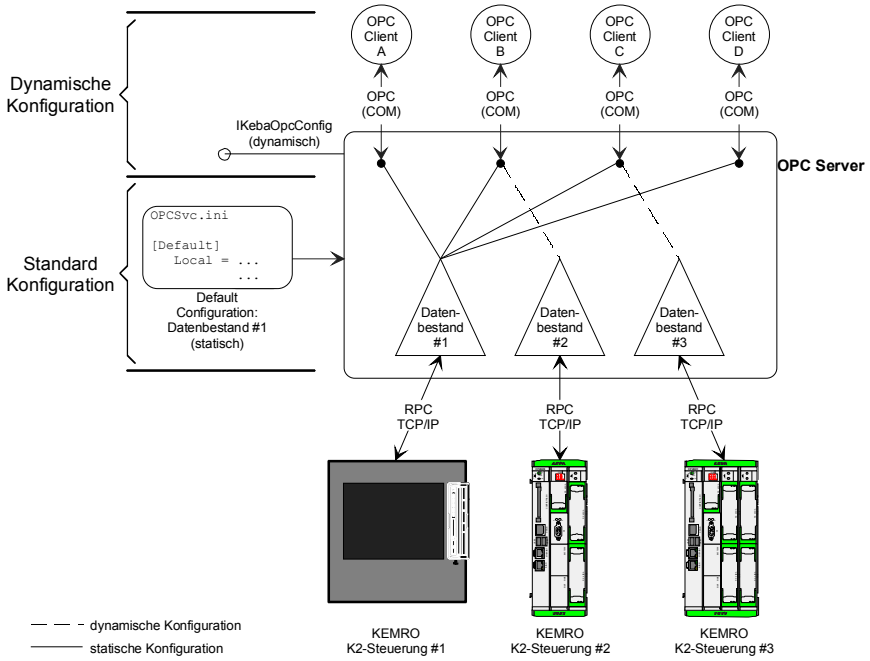


Abb. 1: E-factory.Basecom (Überblick)

e-factory.Basecom bietet eine OPC Schnittstelle gemäß den Spezifikationen "OPC Data Access Custom Interface Version 2.04". OPC Spezifikationen ab der Version 2.00 werden unterstützt, ältere Versionen nicht.

Über die in der OPC Spezifikation definierte Automation Schnittstelle ist der Zugriff auf den E-factory.Basecom auch von Visual Basic und Visual Basic for Applications (Access, Excel,...) aus möglich.

Fordert ein OPC Client eine Verbindung zu einer Steuerung über e-factory.Basecom an, so liest dieser seine gespeicherte Standard-Konfiguration (OPCSvc.ini) ein und konfiguriert sich entsprechend. Damit steht eine Standard-Konfiguration der EC/CC200 Steuerung und ein "Default User" zur Verfügung (siehe Kap. "Standard-Konfiguration"). Mit Hilfe dieser voreingestellten Konfigurationsdaten kann sich ein Standard OPC Client zu vordefinierten Steuerungen als Standardbenutzer verbinden. Zusätzlich ist es über die dynamische Konfiguration möglich, von einem OPC Client Verbindung zu weiteren Steuerungen aufzunehmen (IKebaOpcConfig).

Nach erfolgreicher Herstellung der Verbindung kann der OPC Client auf den Datenbestand der Steuerung zugreifen.
Voraussetzung für eine Datenverbindung zu einer EC/CC200-Steuerung ist jedoch, dass die betreffende Maschine/Steuerung für den Betrieb mit e-factory lizenziert ist.

Funktionalitäten der EC/CC200-Steuerung

Folgende Funktionalitäten stehen über den OPC-Server zur Verfügung:

- Abfragen ("Browsen"), Lesen und Schreiben von Systemvariablen
- Lesen von Systemvariablen-Attributen (Kurz-, Langtext, Format, Einheit,...)
- Lesen und Quittieren von Alarmen
- Verwalten von Benutzern (An- und Abmeldung) und deren Rechte
- Unterstützung mehrerer Sprachen (Sprachumschaltung)
- Lesen von Systemereignissen (Infolog)
- Dynamische Konfiguration des OPC-Servers (Hinzufügen von Steuerungen mittels IKebaOpcConfig-Schnittstelle)

Konfiguration des OPC-Servers

Unter der Konfiguration eines **E-factory.Basecom** wird die Menge der Steuerungen verstanden, auf die ein OPC Client Zugriff hat. Pro OPC Client gibt es eine Konfiguration (siehe Abb. 2). Eine Änderung der Konfiguration kann ein OPC Client jederzeit selbständig vornehmen. Diese Funktionalität wird nicht über die OPC Schnittstelle, sondern über das COM Interface `IKebaOpcConfig` angeboten.

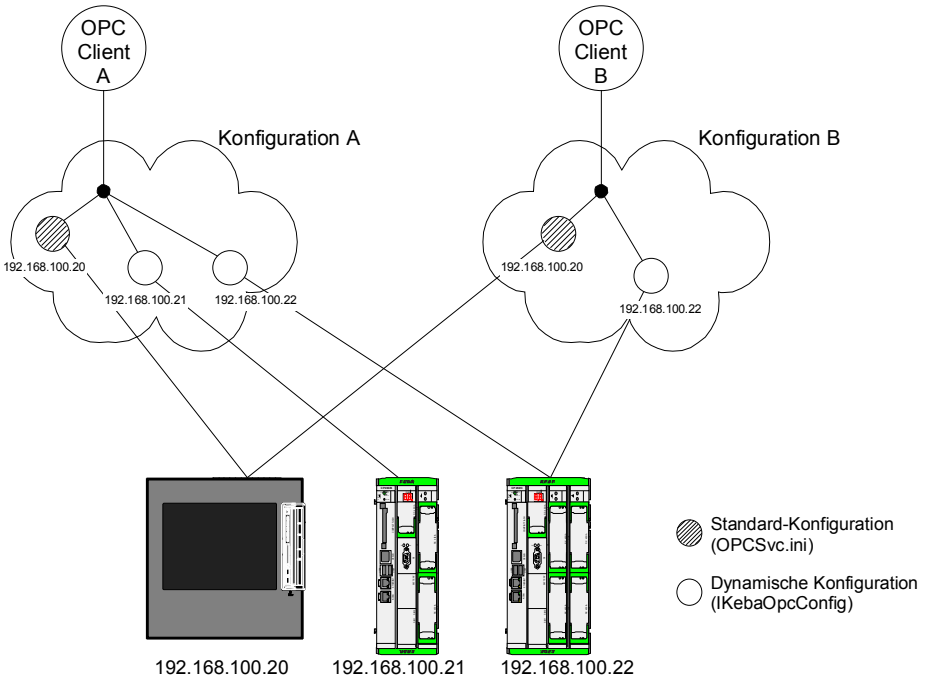


Abb. 2: OPC Clients mit unterschiedlichen Konfigurationen

Eine Konfiguration ist als hierarchische Datenstruktur implementiert, die über die in OPC definierte Browse-Schnittstelle abgefragt werden kann. Jeder Knoten (OPC Item) in dieser Datenstruktur ist über einen Namen eindeutig identifizierbar. Die Namen der Knoten dieser Datenstruktur entsprechen den IP Adressen bzw. den Host Namen der verfügbaren Steuerungen.

Standard-Konfiguration

Der Start von **e-factory.Basecom** erfolgt beim ersten Verbindungsaufbau eines OPC Clients zu **e-factory.Basecom**. Die Standard-Konfiguration (OPCSvc.ini) wird vom Rechner geladen, auf dem der OPC-Server installiert ist und jedem neu hinzukommenden OPC Client zur Verfügung gestellt. In Abbildung 2 ist die Standard Konfiguration von e-factory.Basecom schraffiert dargestellt.
Die Datei "OPCSvc.ini" liegt im Installationsverzeichnis von **e-factory.Basecom**.

Beispiel für eine OPCSvc.ini - Datei:

```
[DefaultHosts]

myBFR = ecpu51                      alle Systemvariablen der Steue-
                                    rung mit dem angegebenen Host-
                                    Namen werden gelesen
myBFR = 192.168.181.52              alle Systemvariablen der Steue-
                                    rung mit dem angegebenen Host-
                                    IP-Adresse werden gelesen
```

Dynamische Konfiguration

Mit Hilfe der COM Schnittstelle `IKebaOpcConfig` kann jeder OPC Client seine Konfiguration individuell – über die Standard Konfiguration hinaus – verändern und anpassen. Es ist für OPC Clients möglich, Steuerungen zur erweiterten Konfiguration hinzuzufügen und auch wieder von der individuellen Konfiguration zu entfernen.

Nähere Informationen siehe:
OPC-Server Schnittstellenbeschreibung, Kapitel "Konfiguration"

Datenverbindung

e-factory.Basecom baut sofort nach dem Start Datenverbindungen zu allen in der Standard Konfiguration befindlichen und mit einer e-factory-Lizenz ausgestatteten EC/CC200-Steuerungen auf.

Eine Datenverbindung von **e-factory.Basecom** zu einer EC/CC200-Steuerung besteht solange diese Steuerung Teil einer Konfiguration mindestens eines OPC Clients ist.

Bricht eine Datenverbindung zu einer Steuerung aus technischen Gründen, die nicht im Bereich von e-factory.Basecom liegen, ab, so wird von e-factory.Basecom versucht, diese Datenverbindung wiederherzustellen. Nach einem erfolgreichen Wiederherstellen einer Datenverbindung können die davon betroffenen OPC Clients sofort wieder auf die entsprechenden Steuerungen zugreifen.
Dieser Zugriff funktioniert aber nur, wenn sich die Systemvariablen nicht geändert haben (OPC-Server aktualisiert den Browse-Baum nicht automatisch) und die Steuerung inzwischen nicht neu gestartet wurde.

3 Funktionalität

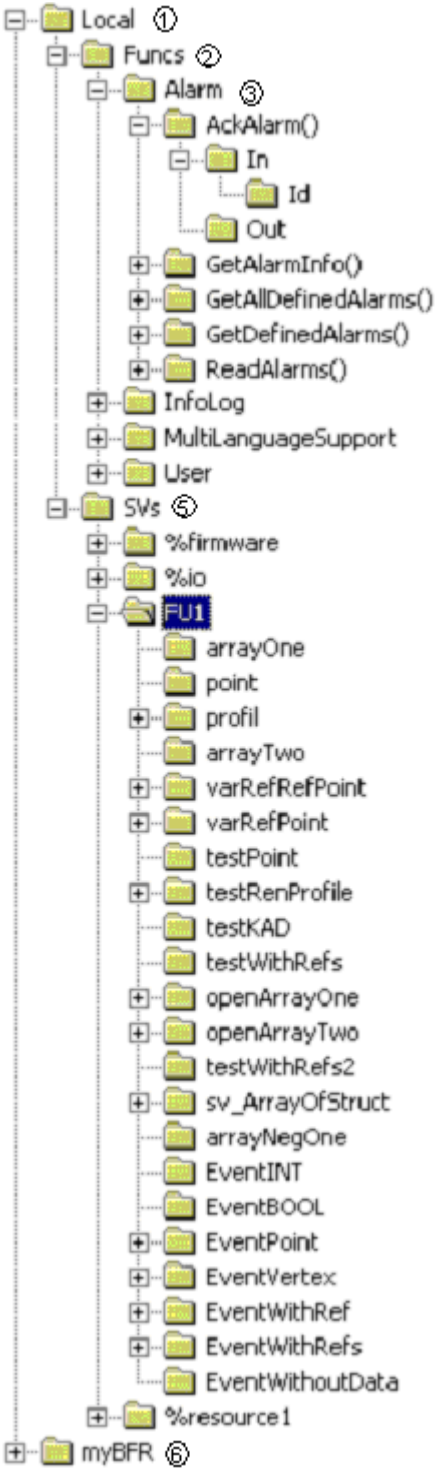
OPC Items

Die gesamte Funktionalität – mit Ausnahme der Konfiguration – und der gesamte Datenbestand von e-factory.Basecom wird über OPC Items abgebildet.

Die Systemvariablen (siehe Abb. 3 Nr. ⑤) werden in einem Baum von OPC-Items abgebildet.
(siehe Kap. "Anlegen von Systemvariablen)

Ändert sich die Menge und/oder die Struktur der Systemvariablen auf der Steuerung, so wird diese Änderung nicht in e-factory.Basecom nachgezogen, da der OPC-Server den Browse-Baum nicht automatisch aktualisiert. Die Änderungen stehen erst allen OPC Clients nach manueller Aktualisierung zur Verfügung.

Wie aus dem Baum in Abb. 3 ersichtlich, werden auch KEBA spezifische Funktionen (z.B. Alarme Abb. 3 Nr. ③, Infolog, usw.) unterstützt.



① Name der 1. konfigurierten Steuerung
② Verzeichnis der Funktionen
③ Alarmfunktionen
⑤ Systemvariablen
⑥ Name der 2. konfigurierten Steuerung

Abb. 3: Verzeichnisstruktur

4 EC/CC200 spezifische Erweiterungen

Über die Standardfunktionalität hinaus, gibt es noch eine Vielzahl von EC/CC200 spezifischen Erweiterungen.

OPC Item Properties

Alle im Systemvariablen Editor (Bestandteil von KEMRO.iecedit) spezifizierten SV Attribute werden als OPC Vendor Properties an den OPC Client weitergereicht.
Mit der Funktion `QueryAvailableProperties()` kann ausgelesen werden, welche Attribute für eine Systemvariable vergeben wurden. In einem Array werden von der Funktion die verfügbaren Attribute als `PropertyIDs()` (definierte Nummern) retourniert:

PropertyID	Bezeichnung	
9001	long text	
9002	short text	
9005	alarm text	
9010	format name	
9011	format type:	
	1: FormatTypeNumber	2: FormatTypeString
	3: FormatTypeText	4: FormatTypeSymbol
9012	format filename	
9013	format precomma ISO absolut	
9014	format postcomma ISO absolut	
9015	format precomma Imperial absolut	
9016	format postcomma Imperial absolut	
9017	format precomma ISO relativ	
9018	format postcomma ISO relativ	
9019	format precomma Imperial relativ	
9020	unit name	
9021	unit filename	
9022	unit transformation class	
9023	unit divisor	
9024	unit multiplier	
9028	format postcomma Imperial rel.	
9029	format string length	
9030	vargroup names	
9031	listgroup names	
9040	properties file	
9050	plausibility limits	
9051	plausibility minimum double	
9052	plausibility maximum double	
9053	plausibility minimum long	
9054	plausibility maximum long	
9060	unit name relativ	
9070	display level	
9071	input level	

Mit `GetItemProperties()` können die Daten der Attribute gelesen werden. (siehe Beispiel 5)
Nähere Informationen und Beschreibung der oben angeführten Bezeichnungen: siehe KEMRO.iecedit Onlinehilfe; Attributieren von Systemvariablen

Lesen und Schreiben von Systemvariablen beschränken

In der Konfigurationsdatei `OpcSvc.ini` werden Werte für die Einträge `Input Level` und `Display Level` eingetragen.
Diese Werte werden mit den in den Systemvariablen konfigurierten Werten verglichen.

Ist der Wert des `Input Level` einer Systemvariablen größer als der in der `OpcSvc.ini` konfigurierte Wert, so kann auf diese Systemvariable nicht geschrieben werden (Schreiboperation liefert einen Fehler).
Ist der Wert des `Display Level` einer Systemvariablen größer als der in der `OpcSvc.ini` konfigurierte Wert, so kann der Wert der Systemvariablen nicht gelesen werden (Leseoperation liefert einen Fehler).
Im IEC-Projekt müssen für die manipulationsgefährdeten Systemvariablen entsprechend niedrige Werte gesetzt werden.

Damit die Werte später nicht manipuliert werden können, gibt es für jeden Wert (1 – 16) ein fixes Passwort. Die Liste dieser Passwörter wird mit dem OPC-Server mitgeliefert.

Weitere Informationen: Siehe Kemro.iecedit Onlinehilfe.

Interface IKebaOpcAccessRights

Das Interface `IKebaOpcAccessRights` beinhaltet folgende Methoden zum Sethen und Ermitteln von `DisplayLevel` und `InputLevel`:

`SetInputLevelPassword()`	Setzt das InputLevel-Passwort und liefert den InputLevel, der dem Passwort entspricht.
`GetInputLevel()`	Liefert den aktuellen InputLevel. Dieser kann sich von dem, den SetInputLevel-Password() liefert, unterscheiden, wenn ein höherer Level in der OpcSvc.ini-Datei definiert ist.
`SetDisplayLevelPassword()`	Setzt das DisplayLevel-Passwort und liefert den DisplayLevel, der dem Passwort entspricht.
`GetDisplayLevel()`	Liefert den aktuellen DisplayLevel. Dieser kann sich von dem, den SetDisplayLe-velPassword() liefert, unterscheiden, wenn ein höherer Level in der OpcSvc.ini-Datei definiert ist.

Die Angabe von `DisplayLevel`- und `InputLevel`-Passwörtern hat nur Auswirkung auf die aktuelle Clientsession (= Verbindung zum OPC-Server / COM-Referenz auf den OPC-Server). Die Levels gelten für alle Systeme in der aktuellen OPC-Konfiguration des Clients (statisch und dynamisch).

D.h. wenn sich zu selben Zeit ein anderer Client über den OpcServer zur selben Steuerung verbindet, gelten für diesen die in der `OpcSvc.ini`-Datei oder bei einem eigenen `Set…LevelPassword()`-Aufruf angegebenen `DisplayLevel`- und `InputLevel`-Passwörter.
Wenn sowohl in der `OpcSvc.ini`-Datei als auch durch Aufrufe von `Set…LevelPassword()`-Methoden Passwörter für `DisplayLevel / InputLevel` angegeben wurden, wird für die Zugriffsprüfung immer der höhere Level verwendet.

Hinweis
Am OpcServer mit einer älteren Version als V1.88 muss die Prüfung auf DisplayLevel und InputLevel deaktiviert werden. Beim OpcServer-Setup gibt es eine Option in der Config.ini-Datei, mit der diese Level-Prüfung des OpcServers bereits bei der Installation deaktiviert wird.

Alarme

Durch das Lesen von OPC Items werden die auf einer EC/CC200 Steuerung anstehenden Alarme ausgewertet.
Ein Alarm besteht aus

- Name
- Status
- Zeitpunkt der Auslösung
- Alarmklasse (Priorisierung)
- Alarmtext
- Kennzeichen, das angibt, ob der Alarm im Infolog protokolliert wurde
- Quittierungsart (Quittierung durch den Benutzer und/oder durch die Applikation)
- Alarmzustand (aktiv, inaktiv, quittiert, gelöscht)

Für den Zugriff auf Alarme im EC/CC200-System sind folgende Items verfügbar:

`<hostname>.Funcs.Alarm.AckAlarm()`	Alarm quittieren
`<hostname>.Funcs.Alarm.GetAlarmInfo()`	Anzahl der anstehenden Alarme und den jüngsten Alarm abfragen
`<hostname>.Funcs.Alarm.GetAllDefinedAlarms()`	Ermitteln aller definierten Alarme der SPS (inklusive Maschineneinheiten-Angabe)
`<hostname>.Funcs.Alarm.GetDefinedAlarms()`	Ermitteln aller definierten Alarme einer Maschineneinheit
`<hostname>.Funcs.Alarm.ReadAlarms()`	Auslesen aller anstehenden Alarme

Nähere Informationen:
Siehe EC/CC200 - e-factory.Basecom Schnittstellenbeschreibung; Kapitel "Funktionalität Alarm"

Benutzerverwaltung

Für jeden der Steuerung bekannten Benutzer wird ein Benutzerprofil abgelegt, in dem benutzerspezifische Daten definiert werden.

Die von einem OPC Client durchführbaren Aktionen sind:

<hostname>.Funcs.User.AckLoginRequest()	Bestätigen einer Login Anfrage
<hostname>.Funcs.User.AddUser()	Hinzufügen eines neuen Benutzers zum System
<hostname>.Funcs.User.CancelLogin()	Abbrechen eines laufenden Anmeldeversuchs
<hostname>.Funcs.User.GetLoginRequests()	Abfragen aller anstehenden Anmeldeversuche
<hostname>.Funcs.User.GetLoginStatus()	Abfragen des momentanen Status des Anmeldevorganges
<hostname>.Funcs.User.GetSession()	Informationen über die momentan laufende Sitzung abfragen
<hostname>.Funcs.User.GetSessions()	Status der momentan laufenden Sitzungen abfragen
<hostname>.Funcs.User.GetUserAttributes()	Statische Attribute eines registrierten Benutzers abfragen
<hostname>.Funcs.User.GetUserConfig()	Benutzerkonfiguration der Steuerung auslesen
<hostname>.Funcs.User.GetUserNames()	Namen aller registrierten Benutzer auslesen
<hostname>.Funcs.User.GetWriteAccessSessions()	Anzahl der angemeldeten Sitzungen mit Schreibzugriff auslesen
<hostname>.Funcs.User.IsLocalStation()	Abfrage des Stationsstatus (Remote oder Lokal)
<hostname>.Funcs.User.IsValidPassword()	Gültigkeit des Passwortes prüfen
<hostname>.Funcs.User.LogIn()	Anmelden eines Benutzers beim System
<hostname>.Funcs.User.LogOut()	Abmelden eines Benutzers beim System
<hostname>.Funcs.User.LogOutAndModify()	Anmelden eines Benutzers beim System und gleichzeitiges Ändern des Profils
<hostname>.Funcs.User.ModifyPassword()	Ändern des Passwortes
<hostname>.Funcs.User.ModifyUser()	Ändern der Daten eines Benutzers
<hostname>.Funcs.User.RemoveUser()	Entfernen eines Benutzers vom System
<hostname>.Funcs.User.TerminateUserSessions()	Beenden aller laufenden Sitzungen des angegebenen Benutzers

Nähere Informationen:
Siehe EC/CC200 - E-factory.Basecom Schnittstellenbeschreibung; Kapitel "Funktionalität User"

Sprachumschaltung

In dem Benutzerprofil der Steuerung wird die Information über die verwendete Sprache des Benutzers abgelegt.

Folgende Teile sind mehrsprachig ausgeführt und werden dem angemeldeten Benutzer in seiner Sprache angeboten:

- Kurz– und Langtext der Systemvariablen,
- Alarmtexte
- Infolog-Texte.

Verfügbare Items:

`<hostname>.Funcs.MultiLanguageSupport.SetDefaultLanguageId()`	Standardmäßig voreingestellte Sprache angeben
`<hostname>.Funcs.MultiLanguageSupport.SynchronizeMultiLanguageSupport()`	Synchronisieren der Property-Files zwischen E-factory.Basecom Cache und KEMRO Steuerungen
`<hostname>.Funcs.MultiLanguageSupport.InitAlarmDefinitionsContainer()`	Initialisieren des OPC-Server internen Alarm-Definitions-Containers
`<hostname>.Funcs.MultiLanguageSupport.InitUnitDefinitionsContainer()`	Initialisieren des OPC-Server internen Einheiten-Definitions-Containers
`<hostname>.Funcs.MultiLanguageSupport.InitSVTextDefinitionsContainer()`	Initialisieren des OPC-Server internen SVText-Definitions-Containers
`<hostname>.Funcs.MultiLanguageSupport.GenerateAlarmText()`	Alarmtexte manuell generieren
`<hostname>.Funcs.MultiLanguageSupport.GetAlarmDefinition()`	Abfragen des definierten Alarmtext
`<hostname>.Funcs.MultiLanguageSupport.GetUnitInformation()`	Unit-Info zu einem Unit-Namen ermitteln
`<hostname>.Funcs.MultiLanguageSupport.GetPureSVText()`	Abfragen des definierten Lang- und Kurztextes inkl. Trennzeichen zwischen Lang- und Kurztext

Nähere Informationen:
Siehe EC/CC200 - e-factory.Basecom Schnittstellenbeschreibung; Kapitel "Funktionalität MultiLanguageSupport"

Infolog

Im Infolog werden Systemereignisse (Alarme, Benutzerwechsel, Systemfehler, usw.) aufgezeichnet bzw. protokolliert.

Durch das Lesen von OPC Items werden jeweils max. 100 Infolog Einträge von einer K2-Steuerung zum OPC Client übertragen.
Erst wenn alle alten Einträge gelesen wurden, werden mit jedem neuerlichen Einlesevorgang die seit dem letzten Lesen auf der Steuerung neu hinzugekommenen Einträge gelesen.

Verfügbare Items:

`<hostname>.Funcs.InfoLog.ReadEntries()`	Lesen der Infolog-Einträge

Nähere Informationen:
Siehe EC/CC200 - e-factory.Basecom Schnittstellenbeschreibung; Kapitel "Funktionalität Infolog"

5 Beispiele

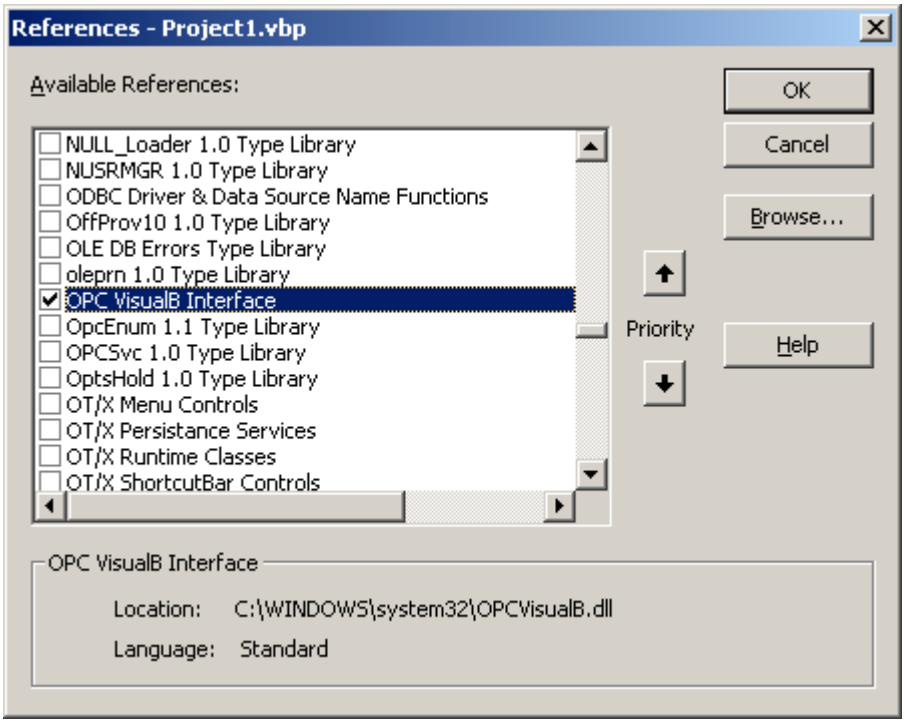
Voraussetzungen

1. Erfolgreiche Installation von e-factory.Basecom
2. OPCVisualB.dll ist registriert und in VB referenziert (siehe unten)

Die `OPCVisualB.dll` wird bei der Installation des OPC-Servers in das OPC-Installationsverzeichnis kopiert.

Referenzierung der OPCVisualB.dll für Excelmacros:

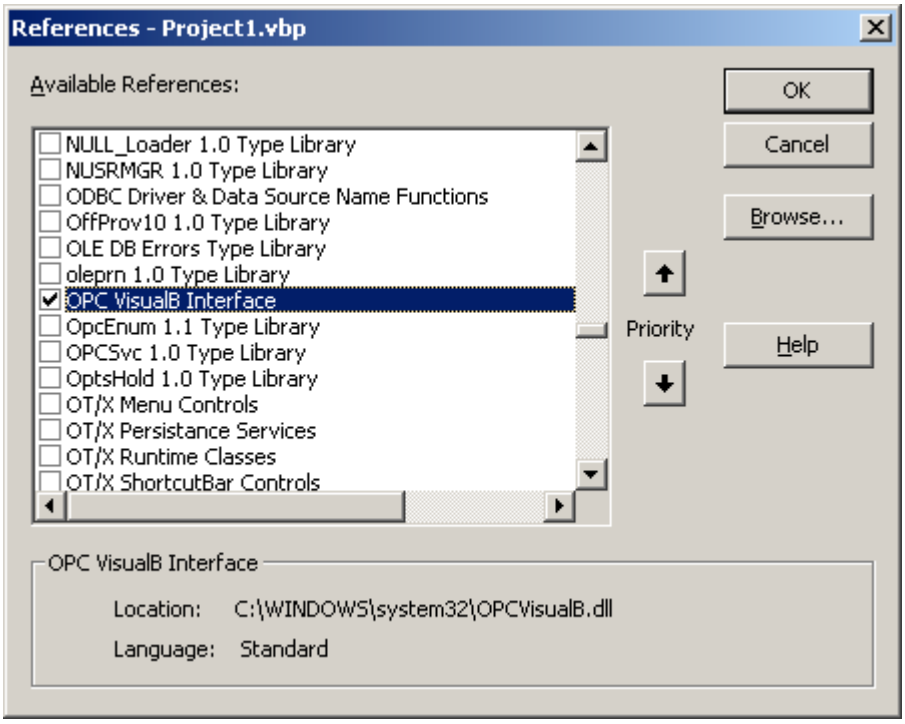
1. Visual Basic Editor aktivieren (Im Excel unter Extras -> Makro -> Visual-Basic- Editor
2. Unter Extras –> Verweise überprüfen, ob die Checkbox bei "OPC VisualB Interface" aktiviert ist und der richtige Pfad eingestellt ist.



Screenshot Verweisfenster (Excel), "OPC VisualB Interface" aktiviert

Referenzierung der OPCVisualB.dll für Visual Basic:

Unter Project -> References überprüfen, ob die Checkbox bei "OPC VisualB Interface" aktiviert ist und der richtige Pfad eingestellt ist.



Screenshot References-Fenster (Visual Basic), "OPC VisualB Interface" aktiviert

Beispiel 1: Funktionsaufruf

Ein Funktionsaufruf wird wie folgt durchgeführt:

1) Falls die Funktion Eingangsparameter benötigt, so müssen diese beschrieben werden
2) Beliebigen Wert auf die Funktion schreiben und dadurch die Funktion aufrufen.
3) Ergebnisse in den Ausgangsparametern auslesen

Prüfen eines Passwortes:

Eingangsparameter:

OPC Item: `Local.Funcs.User.IsValidPassword().In.Name`
Wert: „Testbenutzer“
Beschreibung: Mit dieser Funktion wird der Name in "`Name`" geschrieben.

OPC Item: `Local.Funcs.User.IsValidPassword().In.Password`
Wert: „Testpasswort“
Beschreibung: Mit dieser Funktion wird der Name in "`Name`" geschrieben.

Aufruf der Funktion:

OPC Item: `Local.Funcs.User.IsValidPassword()`
Wert: 0
Beschreibung: Diese Funktion muss mit einem beliebigen Wert beschrieben werden um diese damit aufzurufen.

Ausgangsparameter auslesen:

OPC Item: `Local.Funcs.User.IsValidPassword().Out.IsValid`
Wert: 0
Beschreibung: Diese Funktion gibt an, ob das Paar Name und Passwort an der verbundenen Steuerung gültig ist.

Beispiel 2: Synchrones Lesen und Schreiben von Systemvariablen

In diesem Beispiel werden folgende Punkte behandelt:

- Verbinden mit e-factory.Basecom
- Anlegen einer OPC Group
- Anlegen einiger OPC Items
- Synchrones Lesen und Schreiben von Systemvariablen

```
Private Const OPCCache = 1
Private Const OPCDevice = 2

  Dim server As OPCServer
  Dim Groups As OPCGroups
  Dim Items As OPCItems
  Dim group1 As OPCGroup

  Rem Instanzieren eines neuen OPCServer Objekts
  Set server = New OPCServer

  Rem Verbinden mit einem E-factory.Basecom
  server.Connect cServername

  Rem Auslesen des Hersteller-Informationsstrings
  MsgBox ("Info:" + server.VendorInfo)

  Rem OPCGroups Collection des Servers ...
  Set Groups = server.OPCGroups

  Rem neue Gruppe namens "Group1" zu den OPCGroups des Servers hin-
zufügen
  Set group1 = Groups.Add("Group1")

  Rem OPCItems Collection der neuen Gruppe ...
  Set Items = group1.OPCItems

  Rem benötigte Variablen für AddItems reservieren
  Dim ItemIds(3) As String
  Dim ClientHandles(3) As Long
  Dim ServerHandles() As Long
  Dim Errors() As Long

  Rem ItemIds und ClientHandles zuweisen
  ItemIds(1) = "Local.SVs.FU1.point.x"
  ClientHandles(1) = 100
  ItemIds(2) = "Local.SVs.FU1.point.y"
  ClientHandles(2) = 101
  ItemIds(3) = "Local.SVs.FU1.point.z"
  ClientHandles(3) = 102

  Rem Items zur OPC Group "Group1" hinzufügen
  Items.AddItems 3, ItemIds, ClientHandles, ServerHandles, Errors

  Rem Fehler Array abfragen
  For i = 1 To 3
    If Errors(i) <> 0 Then
```

```
        MsgBox "Incorrect Access Path specified, Unable to add I-
tems", vbCritical, "OPCReadWrite"
        Exit Sub
    End If
  Next
Rem Array of Variant für gelesene Werte deklarieren
  Dim valuesread() As Variant

  Rem Synchronen Read durchführen
  group1.SyncRead OPCDEVICE, 3, ServerHandles, valuesread, Errors

  Rem Fehler Array abfragen
  For i = 1 To 3
    If (Errors(i) <> 0) Then
      MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) + " Er-
ror!")
    Else
      MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) + " Va-
lue: " + Str(valuesread(i)))
    End If
  Next

  Rem Array of Variant für zu schreibende Werte anlegen
  Dim valueswrite(3) As Variant

  Rem Wertearray belegen


  valueswrite(1) = 17;
  valueswrite(2) = 35;
  valueswrite(3) = 99;

  Rem Synchronen Write durchführen
  group1.SyncWrite 3, ServerHandles, valueswrite, Errors

  Rem Fehler Array abfragen
  For i = 1 To 3
    If (Errors(i) <> 0) Then
      MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) +  "Er-
ror!")
    Else
      MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) + "  +
"OK!")
    End If
  Next

  Rem Serververbindung abbauen
  server.Disconnect

  Rem Objekte freigeben
  Set group1 = Nothing
  Set browser = Nothing
  Set Items = Nothing
  Set Groups = Nothing
  Set server = Nothing
```

Beispiel 3: Asynchrones Lesen von Systemvariablen

In diesem Beispiel werden folgende Punkte behandelt:

- Asynchrones Lesen einer OPC Group
- OPC Group für asynchrone Benachrichtigungen (Events) freigeben
- Erstellung einer Eventbehandlungs-Routine

Damit die Gruppe asynchrone Benachrichtigungen erhalten kann, muss sie sich erst beim Server anmelden. In VB bedeutet das, dass die Eigenschaft `IsSubscribed` der Gruppe auf TRUE gesetzt werden muss.
Weiters muss bei der Deklaration der Gruppe `group1` das Schlüsselwort `WithEvents` verwendet werden, damit die OPCGroup als Objekt, das auf Events reagieren soll, gekennzeichnet ist. Das Schlüsselwort `WithEvents` darf nur für Klassenobjekte verwendet werden.

Die eindeutige Zuordnung eines `AsyncReadComplete` – Events zu einem bestimmten `AsyncRead` – Aufruf erfolgt mittels `ClientTransactionID`, welche vor dem `read` – Befehl vom Client gesetzt wird. Beim `read` – Aufruf wird eine `ServerTransactionID` zurückgegeben, die benötigt wird um eine gerade laufende asynchrone Operation abzubrechen (`OPCGroupAsyncCancel`).

```
Dim WithEvents group1 As OPCGroup

Sub testAsyncRead()
...
Rem Gruppe beim Server für asynchrone Benachrichtigungen anmelden
group1.IsSubscribed = True
...

Rem Variable für Server-generierte TransactionID reservieren
Dim lngServerTransactionID As Long
Rem Client TransactionID setzen (beliebiger Wert grösser 0)
lngClientTransactionID = lngClientTransactionID +1

Rem Asynchronen Read anstossen
group1.AsyncRead 3, ServerHandles, Errors, lngClientTransactionID,
lngServerTransactionID

Rem Fehler Array abfragen
For i = 1 To 3
    If Errors(i) <> 0 Then
        MsgBox "Error at AsyncRead", vbCritical, "OPCReadWrite"
        Exit Sub
    End If
Next

MsgBox("asnyc. Read " + Str(lngClientTransactionID) + " in progress
...")

Rem Warteschleife
While blnReadComplete <> True
    prcTimeOut
Wend
```

```
blnReadComplete = False

Rem Verbindungsabbau und Objekte freigeben
...

End Sub


Private Sub prcTimeOut()

Rem Diese Funktion wartet eine Sekunde
Dim vntStart     As Variant
Dim lngCount     As Long
Dim lngPauseTime As Long

    lngCount = 0
    lngPauseTime = 1
    vntStart = Timer
    Do While Timer < vntStart + lngPauseTime
       Rem andere Prozesse bearbeiten ...
       DoEvents
       lngCount = Timer - vntStart
    Loop

End Sub

Private Sub group1_AsyncReadComplete(ByVal TransactionID As Long,
ByVal NumItems As Long,
ClientHandles() As Long, ItemValues() As Variant, Qualities() As
Long, TimeStamps() As Date,
Errors() As Long)

    Rem Eventbehandlungsroutine für group1

    MsgBox ("Async Read completed - TransactionId: " +
Str(TransactionID))
    For i = 1 To NumItems
MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) + " Value: " +
Str(ItemValues(i)))
    Next

    blnReadComplete = True
End Sub
```

Beispiel 4: Reaktion auf Wertänderung der Items

In diesem Beispiel werden folgende Punkte behandelt:

- Erstellung einer Eventbehandlungs-Routine, die auf Wertänderungen von Items reagiert

Das OPCGroups – Objekt, welches alle Gruppen des Servers enthält, löst für alle aktiven Gruppen (falls sie am Server für eine asynchrone Benachrichtigung registriert sind) einen Event aus, wenn Items der Gruppen ihren Wert ändern.
Wenn eine Gruppe angelegt ist, ist diese standardmäßig aktiv, damit nicht immer alle Gruppen Änderungsevents senden, kann eine Gruppe mit der Eigenschaft `IsActive = FALSE` inaktiv gesetzt werden.
Für dieses Beispiel müssen auch OPCGroups mit dem Schlüsselwort WithEvents deklariert werden.

```
Dim WithEvents Groups As OPCGroups
...
Private Sub Groups_GlobalDataChange(ByVal TransactionID As Long, By-
Val GroupHandle As Long, ByVal NumItems As Long, ClientHandles() As
Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As
Date)

Rem Eventbehandlungsroutine für Wertänderungen

    MsgBox ("Item Values in Group " + Str(GroupHandle) + " changed)
    For i = 1 To NumItems
MsgBox ("ClientItemHandle: " + Str(ClientHandles(i)) + " Value: " +
Str(ItemValues(i)))
    Next

End Sub
```

Beispiel 5: Verwendung der Browser-Schnittstelle

In diesem Beispiel werden folgende Punkte behandelt:

- Browse-Baum des OPC-Servers rekursiv durcharbeiten
- alle Itemnames ausgeben

```
Dim browser As OPCBrowser

  Rem Browser Objekt des Servers abfragen
  Set browser = server.CreateBrowser

  Rem browsen starten ...
  browse browser, ""
...

Sub browse(browser As OPCBrowser, branch As String)

   If (branch <> "") Then
      browser.MoveDown (branch)
   End If

   Rem Blätter im aktuellen Zweig ausgeben ...
   browser.ShowLeafs
   For k = 1 To browser.Count
      MsgBox browser.GetItemID(browser.Item(k))
   Next

   Rem In jeden Unterzweig hineinbrowsen ...
   browser.ShowBranches
   For i = 1 To browser.Count
     browse browser, browser.Item(i)
   Next

   Rem Ausgangszustand wiederherstellen
   If branch <> "" Then browser.MoveUp
   browser.ShowBranches

End Sub
```

Beispiel 6: Lesen von OPC Properties (Systemvariablen-Attibuten)

In diesem Beispiel werden folgende Punkte behandelt:

- Abfrage der verfügbaren Properties
- Auslesen der Eigenschaften der Systemvariablen
- Fehlerprüfung und Ausgabe der Properties

```
' Deklaration der benötigten Variablen
Dim ItemId As String
Dim ItemCount As Long
Dim PropIds() As Long
Dim Descriptions() As String
Dim DataTypes() As Integer
Dim PropValues() As Variant
Dim ErrorArray() As Long
Dim i As Long


' Systemvariable, von der die Eigenschaften gelesen werden sollen
ItemId = "BFR.SVs.system.sv_D1ForceCutting"

' Abfrage der verfügbaren Eigenschaften (Property IDs) der SV
server.QueryAvailableProperties ItemId, ItemCount, PropIds, Descrip-
tions, DataTypes

' Auslesen der Eigenschaften der SV
server.GetItemProperties ItemId, ItemCount, PropIds, PropValues, Er-
rorArray

For i = 1 To ItemCount
  If ErrorArray(i) = 0 Then  ' Prüfung auf Fehler
    MsgBox (Descriptions(i) + ":" + CStr(PropValues(i)))  ' Ausgabe
der Eigenschaften
  Else
    MsgBox ("Error getting Property " + Descriptions(i))  ' Fehler-
meldung
  End If
Next

End Sub
```