

Universelle Visualisierung für den EIB ohne Lizenzkosten

Frank Schlaps

Schlaps & Partner GmbH, Reichelsheim

Der EIB hat sich längst etabliert. Aber an Visualisierungen wagen sich nur wenige heran. Meistens scheidet es an den hohen Einstiegskosten oder an der langwierigen Einarbeitung. Der folgende Artikel beschreibt eine einfache und vor allem kostengünstige Lösung zur Visualisierung beliebig großer EIB-Installationen. Kernstück ist die universelle Schnittstelle EIB.VB von Schlaps & Partner.

Bisherige Visualisierungsansätze:

Um den EIB zu visualisieren, sind entweder relativ teure Kopier- oder Einzellizenzen erforderlich. Oft verteuern notwendige Optionen, die zugekauft werden müssen, das Projekt unerwartet. Einfache Visualisierungen sind zwar leicht zu programmieren, lassen jedoch keine komplexen Funktionen zu. Manche Visualisierungen sind in der Anzahl verwendbarer Prozeßpunkte oder Seiten begrenzt. Durch diese Einschränkungen ist der Kreativität des Programmierers schnell ein Ende gesetzt. Komplexe Visualisierungen verfolgen den Ansatz, daß zwar fast alles möglich ist, jedoch jede Funktion zu Fuß programmiert werden muß (Bekannt als Script, Macro oder Sequenz). Manche Visualisierungen versprechen die Integrierbarkeit aller Bussysteme, bedienen jedoch den EIB mangelhaft. Häufig werden nur Gruppenadresszugriffe statt Kopplungen auf Objektebene unterstützt. Einen völlig anderen Ansatz verfolgt EIB.VB mit dem lizenzkostenfreien Aufbau auf Basis von Visual Basic. Erforderlich sind lediglich das Standardprogramm Visual Basic von Microsoft und ein EIB-Busankoppler von Schlaps & Partner. Wahlweise kann auch der EIB-Zugriff direkt aus Microsoft-Office-Programmen



wie Excel oder Access unter VBA realisiert werden. Die EIB-Ankoppelung erfolgt seriell und funktioniert auf Gruppenadress- oder Objektebene ohne Telegrammverlust bis 100% Buslast. Es werden alle möglichen 32767 Gruppenadressen des EIB im Gruppenadress- und Objektmodus unterstützt. Eine Begrenzung auf bestimmte Seiten pro Projekt oder Prozeßpunkte pro Seite besteht nicht.

Wie kommuniziert der EIB mit Visual Basic?

Im Lieferumfang von EIB.VB ist ein Programm (EIB-Explorer) enthalten, der direkt aus dem ETS-Projekt (ETS = EIB-Tool-Software, das Programm, mit dem der Elektriker den EIB programmiert) die erforderlichen Prozeßpunkte erzeugt. Notfalls können die benutzten Gruppenadressen auch direkt vom EIB aufgezeichnet und von Hand im Explorer eingegeben werden (Notlösung, falls kein ETS-Projekt vorhanden ist). (Bild 1)

Was ist EIB.VB?

EIB.VB besteht aus einem sehr schnellen seriellen Busankoppler (EIB-AT), der eine Filtertabelle und eine EIS-Tabelle enthält, sowie einer

Softwareschnittstelle zu Visual Basic. Die Filtertabelle verhindert, daß unerwünschte Telegramme den PC belasten. Die integrierte EIS-Tabelle (EIS = EIB-Interworking-Standard) ordnet den Datenpunkten die jeweilige Datenpunktart (Bit, Ganzzahl, Kommazahl) zu, sodaß z.B. eine Temperatur als lesbarer Wert und nicht als kryptische Folge von Bytes übertragen wird. Der Busankoppler kommuniziert ohne aufwändige Handshakes mit dem PC über die RS232-Schnittstelle. Fehlt dem PC die serielle Schnittstelle, so ist EIB-AT mit Hilfe eines RS232-USB-Adapters anschließbar. Statt das rechnerintensive EIB-Protokoll zu benutzen, unterhält sich der PC über ASCII-Sequenzen (vom Mensch lesbare Textbefehle) mit dem EIB. Ist die Visualisierung erstellt, so erhält der Endkunde wahlweise ein lauffähiges 32Bit-Programm (*.exe) und kann keine Änderungen vornehmen oder er bekommt ein VB-Programm im Quellcode geliefert und kann es beliebig selbst anpassen. Besitzt der Endkunde selbst Programmierkenntnisse, so kann er sich anhand der EIB-Prozeßpunkte eigenständig eine Visualisierung programmieren.

Benötigt der EIB Statustelegramme?

EIB.VB führt ein objektbasiertes Prozeßabbild. Das bedeutet, daß jedes EIB-Objekt (z.B. Schalt- oder Dimmkanal) eines EIB-Aktors automatisch emuliert wird. Dadurch nehmen alle Prozeßpunkte ohne Statustelegramme auch bei Verwendung von Zentralfunktionen den aktuellen Zustand des Aktors an.

Wie werden Prozeßpunkte programmiert?

Es sind nur drei Befehle erforderlich. $x = \text{EIB.Value}(„0/0/1“)$ gibt einen Wert vom EIB, $\text{EIB.Value}(„0/0/1“) = x$ schreibt einen Wert auf den EIB und $\text{EIB.read}(„0/0/1“)$ fordert von einem EIB-Gerät einen Wert an.

Dipl.-Ing. Frank Schlaps (42) ist Geschäftsführer der Schlaps & Partner GmbH, Reichelsheim, www.schlaps-automation.de

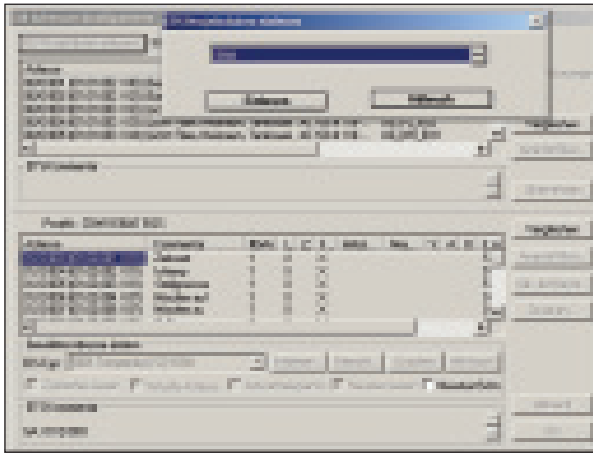


Bild 1. Prozeßpunkterzeugung mit dem EIB-Explorer

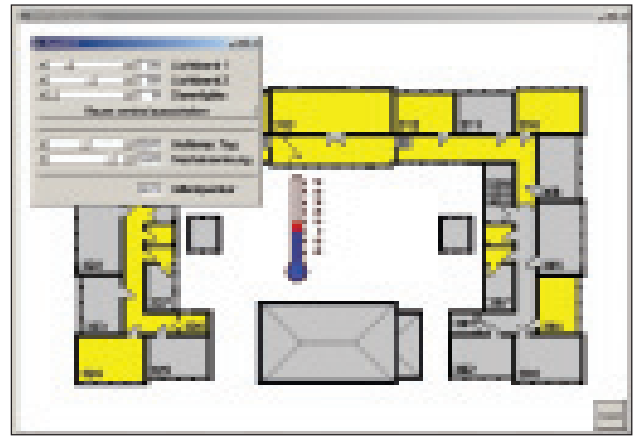


Bild 2. Beispieloberfläche einer Visualisierung mit EIB.VB

Dabei ist völlig gleichgültig, ob es sich um einen Schalter, Dimmwert oder z.B. eine Temperatur handelt. (Bild 2)

Um eine Taste zu programmieren, die ein Licht einschalten soll, platziert man auf der VB-Oberfläche einen Commandbutton (Befehlsschaltfläche), gibt ihm einen Name (z.B. „Licht1 an“) und führt darauf einen Doppelklick aus. Selbständig erzeugt VB den Programmumpf:

```
Private Sub L1_an_Click()
End Sub
```

Aus den mitgelieferten Programmbeispielen ergänzt man:

```
EIB.Value(„0/0/1“) = „1“,
so daß das fertige Unterprogramm wie folgt aussieht:
```

```
Private Sub L1_an_Click()
    EIB.Value(„0/0/1“) = „1“
End Sub
```

Zum Ausschalten schreibt man:
`EIB.Value(„0/0/1“) = „0“`

Soll die selbe Taste das Licht an- und ausschalten, wählt man aus der Programmsammlung das folgende Beispiel aus:

```
Private Sub L1_Click()
    If EIB.Value(„0/0/1“) <> „1“ Then
        EIB.Value(„0/0/1“) = „1“
    Else
        EIB.Value(„0/0/1“) = „0“
    End If
End Sub
```

Weitere Beispiele zum dimmen von Licht, fahren von Rolläden oder einstellen von Temperaturen finden sie unter www.schlaps-automation.de in der Rubrik EIB.VB.

Unterstützt EIB.VB Logikfunktionen?

Dem Programmierer sind fast keine Grenzen gesetzt. Anhand von umfangreichen Programmbeispielen sind neben Logiken relativ einfach Szenen, Zeitfunktionen, Loggings, Statistiken und vieles mehr realisierbar.

Das folgende Beispiel zeigt, wie festzustellen ist, ob noch ein Licht im Haus brennt. Der Einfachheit halber wird hier von drei möglichen Lampen ausgegangen. Es ist eine Oder-Funktion (or) notwendig.

Brennt noch eine der 3 Lampen (0/0/1 bis 0/0/3), so wird die Gruppenadresse 12/0/0 = 1, andernfalls 0.

```
Private Sub Ist_noch_Licht_an()
    If EIB.Value(„0.1.12_1 0/0/1“) = „1“
    or EIB.Value(„0.1.12_2 0/0/2“) = „1“
    or EIB.Value(„0.1.31_0 0/0/3“) = „1“
    Then
        EIB.Value(„12/0/0“) = „1“
    Else
        EIB.Value(„12/0/0“) = „0“
    End If
End Sub
```

In diesem Fall ist es wichtig, daß als Quellen nicht nur Gruppenadressen, sondern Aktorobjekte verwendet werden. Um den Unterschied zu erkennen soll folgendes Beispiel den Aufbau der Objekte zeigen:

Es wird ein Kombiaktor verwendet, an dessen Relais 1 und 2 die Lampen E1 und E2 angeschlossen sind. Die dazugehörigen Schaltobjekte sind demnach 0.0.12_1 und 0.0.12_2. Die dritte Lampe ist an einem anderen Aktor 0.1.31 am Relais 0 ange-

schlossen (entspricht 0.1.31_0).

Es genügt die Betrachtung des Objekts 0.0.12_1.

Dieses belegt folgende Gruppenadressen:

0/0/1, 14/1/0 und 14/0/0.

0/0/1 schaltet das Licht E1 vom lokalen Taster im Raum

14/1/0 ist eine Zentralfunktion und schaltet alle Lampen im Stockwerk

14/0/0 ist eine Zentralfunktion und schaltet alle Lampen im Gebäude (z.B. über eine Zeitschaltuhr).

Wird das Licht E1 lokal eingeschaltet, so nimmt die Gruppenadresse 0/0/1 den Wert 1 an. Schaltet die Zeitschaltuhr dann alle Lampen aus, so ändert sich der Wert der Gruppenadresse 0/0/1 nur dann, wenn man den Wert der Gruppenadresse durch ein Abfragetelegramm erfragt oder die Gruppenadresse 0/0/1 in Form eines Statustelegrams bei Änderung des Objekts automatisch auf den Bus sendet.

Beide Lösungen haben den Nachteil hoher Buslast. Um dies zu vermeiden, verwendet man den Objektmodus.

Dabei erkennt EIB.VB automatisch, daß die Zentralfunktionen (z.B. 14/0/0) mehrere Lampen schalten und aktualisiert damit den Wert der zugehörigen Objekte entsprechend.

Dies spart Buslast und liefert das Ergebnis wesentlich schneller, sodaß die Visualisierung ohne nennenswerte Reaktionszeiten das richtige Bild anzeigt.