

BECKHOFF New Automation Technology

Handbuch | DE

TwinCAT 3

Riedel Communications | RRCS



23.07.2025 | Version: 1.2.0

© Photo: Ralph@Larmann.com

Inhaltsverzeichnis

1	Vorwort.....	5
1.1	Hinweise zur Dokumentation	5
1.2	Zu Ihrer Sicherheit.....	6
1.3	Hinweise zur Informationssicherheit	7
2	Disclaimer	8
3	Übersicht.....	9
3.1	Systemvoraussetzungen	11
3.2	Versionierung	11
4	Installation	12
5	Programmierung	13
5.1	Funktionsbausteine	13
5.1.1	FB_RRCScom.....	13
5.1.2	FB_GetState	14
5.1.3	FB_GetAlive	15
5.1.4	FB_IsConnectedToArtist	16
5.1.5	FB_GetAllLogicSources_V2.....	17
5.1.6	FB_SetLogicSourceState.....	18
5.2	Datentypen	19
5.2.1	ST_LogicSource.....	19
5.2.2	E_RRCS_ErrorCodes	19
6	Beispiele	21
6.1	Tc3_RRCS_Sample01	21
6.2	Tc3_RRCS_Sample02	23
7	Support und Service	28

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.

Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.

Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Kennzeichnungen führen.



EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland.

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.

Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: <https://www.beckhoff.com/trademarks>.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.

Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt



Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem <https://www.beckhoff.de/secguide>.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter <https://www.beckhoff.de/secinfo>.

2 Disclaimer

This publication contains statements about the suitability of our products for certain areas of application. These statements are based on typical features of our products. The examples shown in this publication are for demonstration purposes only. The information provided herein should not be regarded as specific operation characteristics. It is incumbent on the customer to check and decide whether a product is suitable for use in a particular application.

We do not give any warranty that the source code which is made available with this publication is complete or accurate.

THE SAMPLE CODE CONTAINED IN THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED, IMPLIED OR STATUTORY, INCLUDING WITHOUT LIMITATION, ANY WARRANTY WITH RESPECT TO NON-INFRINGEMENT, FREEDOM FROM PROPRIETARY RIGHTS OF THIRD PARTIES OR FITNESS FOR ANY PARTICULAR PURPOSE.

This publication may be changed from time to time without prior notice. No liability is assumed for errors and/or omissions. Our products are described in detail in our data sheets and documentations. Product-specific warnings and cautions must be observed.

For the latest version of our data sheets and documentations visit our website www.beckhoff.de.

© Beckhoff Automation GmbH, January 2025.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization is prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

3 Übersicht

Dieses Beispiel beschreibt die Kommunikationsmöglichkeit zwischen Beckhoff und Riedel Communications.

Riedel Communications

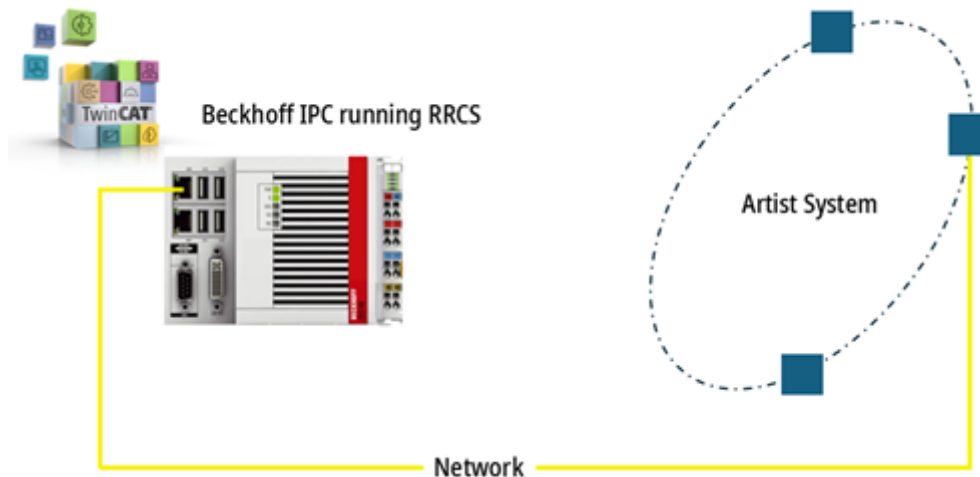
Riedel Communications GmbH & Co. KG entwickelt, fertigt und vertreibt Intercom-, Glasfaser-, Funk- und Audionetzwerksysteme, die in den Bereichen Rundfunk, Veranstaltungen, Theater und Industrie zum Einsatz kommen.



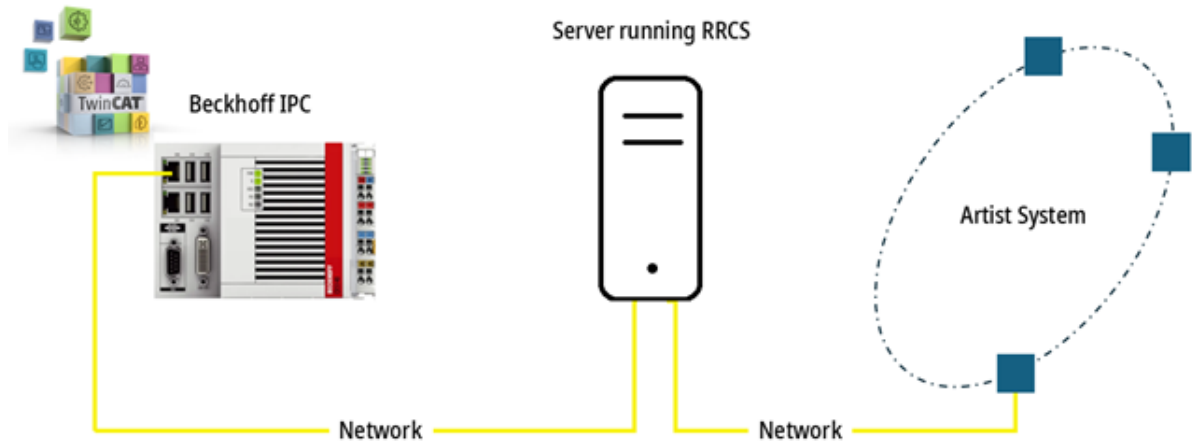
Riedel bietet mit der Riedel Router Control Software (RRCS) ein universelles, XML basierendes Interface, mit dem Intercom-Systeme der Artist Serie gesteuert werden können.

Um RRCS zu nutzen, benötigen Sie eine Installation des RRCS-Servers. Dieser wird auf PCs ausgeführt, die über das Betriebssystem Windows verfügen. Es bieten sich zwei Varianten an:

- Installation des RRCS-Servers auf Beckhoff IPCs, auf denen auch das Programm TwinCAT läuft



- Installation des RRCS-Servers auf einem beliebigen Server



Beckhoff

Beckhoff stellt eine Bibliothek zur Verfügung, mit der Beckhoff-Steuerungssysteme über RRCS mit Riedel Artist Systemen bidirektional kommunizieren können. Diese Bibliothek ist im Weiteren beschrieben und Teil der Beispiele. Die Beispiele können Sie im Kapitel [Beispiel |> 21](#) herunterladen. Um die Bibliothek nutzen zu können, benötigen Sie eine TC1200-Lizenz und eine Lizenz für TF6760. Die Bibliothek und die Beispiele unterliegen dem [Disclaimer |> 8](#) aus Kapitel 2. Zusätzlich werden in dieser Dokumentation zwei beispielhafte Implementierungen aufgeführt.

Zusammenfassung und Benefits

- Kombiniert Beckhoffs offene PC- und EtherCAT-basierte Steuerungstechnik mit Riedel-Kommunikationssystemen.
- Direkte Unterstützung des RRCS-Protokolls von Riedel in TwinCAT
- Ermöglicht die Kommunikation mit Riedel Artist- und Bolero-Systemen sowie die Nutzung von Riedel SmartPanels.
- Erleichtert die dynamische Steuerung von Audiokreisen und die Integration mit Theater- und Show-Ansagesystemen.
- Systemintegratoren können nun offene PC- und EtherCAT-basierte Steuerungstechnikkomponenten von Beckhoff in das Riedel-System integrieren und umgekehrt.
- Beckhoff unterstützt die wesentlichen Teile des RRCS-Protokolls von Riedel in TwinCAT auf Basis von TwinCAT 3 IoT HTTPS/REST (TF6760).

- Dies ermöglicht die Kommunikation mit Riedel Artist- und Bolero-Systemen und die Implementierung von Riedel SmartPanels.
- Basierend auf TwinCAT Speech (TF4500) können mit unseren Text-to-Speech-Funktionsbausteinen Fehlermeldungen oder jede Art von Ansagen erstellt werden. Auch Audiodateien können abgespielt werden.
- Die von TwinCAT Speech (oder anderen Tools) erzeugten Audiodaten können über DANTE an das Riedel-System gesendet werden, indem die virtuelle DANTE-Audiokarte auf Beckhoff Windows 10 oder 11 basierten Systemen verwendet wird.
- Mögliche Anwendungsfälle im Theaterkontext sind das dynamische Ändern und Schalten von Audiokreisen basierend auf der Logik und den Prioritäten des Beckhoff-Steuerungssystems oder, in Kombination mit dem Theater-Bühnenmanagement, der Showsteuerung inklusive Cue-Light-Signalisierung und Durchsagen.

Die Bedienelemente für den Endanwender können aus den folgenden ausgewählt und kombiniert werden:

- Beckhoff Multitouch-Bedienpanels und TwinCAT HMI
- In Kombination mit klassischen Lichtschaltern oder Stellrädern und anderen mechanischen Eingängen
- Riedel SmartPanels oder drahtlose Bolero-Beltpacks als Steuerungs- und Überwachungsschnittstellen

Da das Beckhoff-Steuerungssystem nahezu unbegrenzte Kommunikations- und Steuerungsmöglichkeiten bietet, ist die Steuerung von Audio-/Videosystemen, Bühnen- und Showanwendungen und sogar die Gebäudeautomation mit Temperaturregelung sowie Jalousien und Allgemeinbeleuchtung möglich.

3.1 Systemvoraussetzungen

Um die Beispielprojekte ausführen zu können, müssen folgende Voraussetzungen erfüllt sein:

Technische Daten	Beschreibung
TwinCAT-Version	TwinCAT 3.1 Build 4024.65 oder höher
Erforderliche TwinCAT-Lizenzen	TC1200 TwinCAT 3 PLC TF6760 TwinCAT 3 IoT HTTPS/REST
Riedel Communications RRCS Serversoftware	Informationen hierzu: RRCS

Netzwerktest

Bevor ein Projekt gestartet wird, das die Verwendung eines beliebigen Netzwerkprotokolls vorsieht, muss unbedingt sichergestellt werden, dass das verwendete Netzwerk ordnungsgemäß funktioniert. Wir empfehlen, mit einem einfachen Test zu beginnen, bei dem das Beckhoff-Gerät eine direkte Verbindung zu dem Gerät hat, mit dem es kommunizieren soll. Dies sollte ohne aktivierte Firewalls oder ähnliche Schutzmaßnahmen erfolgen.

1. Überprüfen Sie zunächst die Hardwarekomponenten. Prüfen Sie dazu die Kabelverbindungen und stellen Sie sicher, dass die Router oder Switches eingeschaltet sind und einwandfrei funktionieren.
 2. Überprüfen Sie als nächstes die Netzwerkeinstellungen wie IP-Adressen, Subnetzmasken und Firewall-Einstellungen sowohl auf dem Beckhoff-Gerät als auch auf dem Gerät, mit dem es kommunizieren soll.
- ⇒ Wenn dieser grundlegende Test funktioniert, können Sie die Firewall wie hier beschrieben konfigurieren: https://download.beckhoff.com/download/document/ipc/industrial-pc/ipc_security_guideline_de.pdf

3.2 Versionierung

Datum	Version	Änderung
08.05.2025	3.0.0.0	Erste Version

4 Installation

Installation der Riedel Communications RRCS Software

Je nach gewähltem Aufbau installieren Sie die Software auf dem entsprechenden Gerät. (Siehe Übersicht [► 9]) Der Vorgang ist im RRCS – Quick Startup Guide von Riedel beschrieben.

Installation von TwinCAT

Der Beckhoff Industrie PC muss die Laufzeitumgebung (XAR) installiert haben. Details hierzu finden Sie hier:

https://infosys.beckhoff.com/index.php?content=../content/1031/tc3_installation/179470219.html

Außerdem müssen Sie die TwinCAT Function TF6760 installieren. Details hierzu finden Sie hier:

https://infosys.beckhoff.com/index.php?content=../content/1031/tf6710_tc3_iod_functions/2544553995.html

5 Programmierung

Die bidirektionale Kommunikation zwischen TwinCAT und dem Riedel Communications RRCS-Server erfolgt mittels nachfolgend beschriebenen Funktionsblöcken, die in der Bibliothek Tc3_RRCS enthalten sind.



Um Funktionsblöcke der Bibliothek Tc3_RRCS verwenden zu können, müssen Sie die TwinCAT 3 Function TF6760 IoT HTTPS/REST installieren und lizenzieren!

5.1 Funktionsbausteine

5.1.1 FB_RRCScom



Dieser Funktionsbaustein wird für die Kommunikation mit dem RRCS-Server benötigt



Diesen Funktionsblock sollten Sie nur instanziiert und nicht implizit aufrufen!
Der Aufruf erfolgt mittels Pointer von nachfolgend beschriebenen Bausteinen, z.B. von FB_GetState.

Die IP-Adresse und der Port des RRCS-Servers müssen in der Variablendeklaration festgelegt werden:

```
VAR
fbRRCScom: FB_RRCScom(ipAddr_Server:= '192.168.42.59', ipPort_Server:=8193);
END_VAR
```

Ausgänge

```
VAR_OUTPUT
eCommState      : E_CommState;
nStatusCode      : UINT;
sResponse        : STRING(..);
_pfbSB           : POINTER;
END_VAR
```

Name	Typ	Beschreibung
eCommState	E_CommState	Der Status, in dem sich der Baustein befindet.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status
sResponse	STRING(..)	Zeichenkette mit der unverarbeiteten Antwort des RRCS-Servers. Die maximale Länge der Zeichenkette wird in der Parameterliste der Bibliothek eingestellt.
_pfbSB	POINTER	Nur für interne Verwendung

5.1.2 FB_GetState



Dieser Funktionsblock fragt den Status des RRCS-Servers ab.

Eingänge

```

VAR_INPUT
    bExecute : BOOL;
END_VAR
    
```

Name	Typ	Beschreibung
bExecute	BOOL	Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

Ausgänge

```

VAR_OUTPUT
    eRRCS_ErrorCode : E_RRCS_ErrorCodes;
    bBusy           : BOOL;
    bError          : BOOL;
    nStatusCode     : UINT;
    sGatewayState   : STRING(32);
END_VAR
    
```

Name	Typ	Beschreibung
eRRCS_ErrorCode	E_RRCS_ErrorCodes [► 19]	Errorcode erhalten vom RRCS_Servers.
bBusy	BOOL	Ist TRUE, solange der Baustein mit der Abfrage beschäftigt ist und kein Fehler auftritt.
bError	BOOL	Ist TRUE, wenn bei der Abfrage ein Fehler aufgetreten ist.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status
sGatewayState	STRING(32)	Information des RRCS-Servers, mögliche Zustände: • Working • Standby

FB_GetState verfügt über eine erweiterte FB_init Methode, in der mittels Pointer auf eine Instanz von FB_RRCScom verwiesen werden muss:

```

VAR
    fbGetState: FB_GetState(pfbRRCScom := ADR(fbRRCScom));
END_VAR
    
```


5.1.3 FB_GetAlive



Um sicherzustellen, dass eine korrekte Verbindung zwischen TwinCAT und dem RRCS-Server besteht, kann der Funktionsbaustein FB_GetAlive verwendet werden.

Eingänge

```
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bExecute	BOOL	Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

Ausgänge

```
VAR_OUTPUT
    eRRCS_ErrorCode : E_RRCS_ErrorCodes;
    bBusy           : BOOL;
    bError          : BOOL;
    nStatusCode     : UINT;
    bAlive          : BOOL;
END_VAR
```

Name	Typ	Beschreibung
eRRCS_ErrorCode	<u>E_RRCS_ErrorCodes</u> [► 19]	Errorcode erhalten vom RRCS_Servers.
bBusy	BOOL	Ist TRUE, solange der Baustein mit der Abfrage beschäftigt ist und kein Fehler auftritt.
bError	BOOL	Ist TRUE, wenn bei der Abfrage ein Fehler aufgetreten ist.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status
bAlive	BOOL	Ist TRUE, wenn die Kommunikation zum RRCS-Server funktioniert.

FB_GetAlive verfügt über eine erweiterte FB_init Methode, in der mittels Pointer auf eine Instanz von FB_RRCScom verwiesen werden muss:

```
VAR
    fbGetAlive: FB_GetAlive(pfbRRCScom := ADR(fbRRCScom));
END_VAR
```

5.1.4 FB_IsConnectedToArtist



Dieser Funktionsbaustein wird zur Abfrage verwendet, ob der RRCS-Server mit einem Artist-Geräte kommuniziert.

Eingänge

```
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bExecute	BOOL	Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

Ausgänge

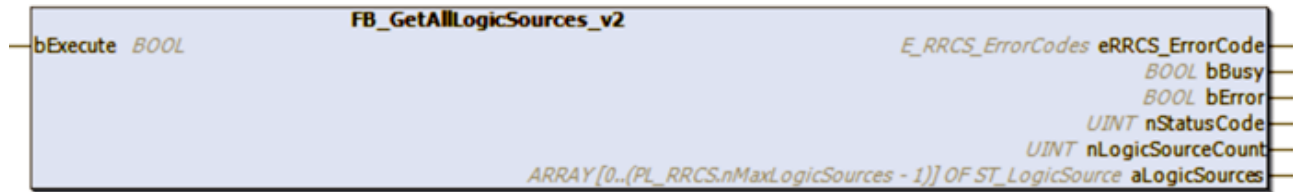
```
VAR_OUTPUT
    eRRCS_ErrorCode : E_RRCS_ErrorCodes;
    bBusy           : BOOL;
    bError          : BOOL;
    nStatusCode     : UINT;
    bIsConnected    : BOOL;
END_VAR
```

Name	Typ	Beschreibung
eRRCS_ErrorCode	E_RRCS_ErrorCodes [► 19]	Errorcode erhalten vom RRCS_Servers.
bBusy	BOOL	Ist TRUE, solange der Baustein mit der Abfrage beschäftigt ist und kein Fehler auftritt.
bError	BOOL	Ist TRUE, wenn bei der Abfrage ein Fehler aufgetreten ist.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status
bIsConnected	BOOL	Ist TRUE, wenn der RRCS-Server mit einem Artist-Gerät kommuniziert.

FB_IsConnectedToArtist verfügt über eine erweiterte FB_init Methode, in der mittels Pointer auf eine Instanz von FB_RRCScom verwiesen werden muss:

```
VAR
    fbIsConnectedToArtist: FB_IsConnectedToArtist(pfbRRCScom := ADR(fbRRCScom));
END_VAR
```

5.1.5 FB_GetAllLogicSources_V2



Dieser Funktionsbaustein fragt die in dem Artist-Gerät definierten 'Logic sources' ab.

Eingänge

```
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Name	Typ	Beschreibung
bExecute	BOOL	Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

Ausgänge

```
VAR_OUTPUT
    eRRCS_ErrorCode : E_RRCS_ErrorCodes;
    bBusy           : BOOL;
    bError          : BOOL;
    nStatusCode     : UINT;
    nLogicSourceCount : UINT;
    aLogicSources   : ARRAY[ ] OF ST_LogicSource;
END_VAR
```

Name	Typ	Beschreibung
eRRCS_ErrorCode	<u>E_RRCS_ErrorCodes</u> [► 19]	Errorcode erhalten vom RRCS_Servers.
bBusy	BOOL	Ist TRUE, solange der Baustein mit der Abfrage beschäftigt ist und kein Fehler auftritt.
bError	BOOL	Ist TRUE, wenn bei der Abfrage ein Fehler aufgetreten ist.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status
nLogicSourceCount	UINT	Die Anzahl der im Artist-Gerät definierten 'Logic Sources'.
aLogicSources	ARRAY[] OF <u>ST_LogicSource</u> [► 19]	Auflistung aller definierten 'Log Sources' in Form eines Arrays. Die maximale Anzahl der Arrayeinträge wird in der Parameterliste definiert.

FB_GetAllLogicSources_V2 verfügt über eine erweiterte FB_init Methode, in der mittels Pointer auf eine Instanz von FB_RRCScom verwiesen werden muss:

```
VAR
    fbGetAllLogicSources_V2: FB_GetAllLogicSources_V2(pfbRRCScom := ADR(fbRRCScom));
END_VAR
```

5.1.6 FB_SetLogicSourceState

FB_SetLogicSourceState		
bExecute	BOOL	E_RRCS_ErrorCodes eRRCS_ErrorCode
nObjectID	UDINT	BOOL bBusy
bState	BOOL	BOOL bError
aLogicSources	ARRAY[0..(PL_RRCS.nMaxLogicSources - 1)] OF ST_LogicSource	UINT nStatusCode

Dieser Funktionsblock bietet die Möglichkeit, den Zustand einer 'Logic Source' zu ändern. Dies geschieht unter Verwendung der eindeutigen ObjectID.

Eingänge

```
VAR_INPUT
    bExecute      : BOOL;
    nObjectID     : UDINT;
    bState        : BOOL;
    aLogicSources : ARRAY[ ] OF ST_LogicSource;
END_VAR
```

Name	Typ	Beschreibung
bExecute	BOOL	Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.
nObjectID	UDINT	Auf 'Logic Sources' wird über die eindeutige ObjectID zugegriffen.
bState	BOOL	Angabe des gewünschten Zustands.
aLogicSources	ARRAY[] OF ST_LogicSource ► 19	Auflistung aller definierten 'Log Sources' in Form eines Arrays. Die maximale Anzahl der Arrayeinträge wird in der Parameterliste definiert.

Ausgänge

```
AR_OUTPUT
    eRRCS_ErrorCode : E_RRCS_ErrorCodes;
    bBusy           : BOOL;
    bError          : BOOL;
    nStatusCode     : UINT;
END_VAR
```

Name	Typ	Beschreibung
eRRCS_ErrorCode	E_RRCS_ErrorCodes ► 19	Errorcode erhalten vom RRCS_Servers.
bBusy	BOOL	Ist TRUE, solange der Baustein mit der Abfrage beschäftigt ist und kein Fehler auftritt.
bError	BOOL	Ist TRUE, wenn bei der Abfrage ein Fehler aufgetreten ist.
nStatusCode	UINT	Der Statuscode wird von Servern zu jeder HTTP-Anfrage als Antwort geliefert. Der Server teilt durch diesen Code dem Client mit, ob die Anfrage erfolgreich war. Im Normalfall wird 200 (OK) zurückgegeben. Detaillierte Informationen hierzu: developer.mozilla.org/de/docs/Web/HTTP/Reference/Status

FB_SetLogicSourceState verfügt über eine erweiterte FB_init Methode, in der mittels Pointer auf eine Instanz von FB_RRCScom verwiesen werden muss:

```
VAR
    fbSetLogicSourceState: FB_SetLogicSourceState_V2(pfbRRCScom := ADR(fbRRCScom));
END_VAR
```

5.2 Datentypen

5.2.1 ST_LogicSource

```

TYPE ST_LogicSource:
STRUCT
    sName          : STRING(32);
    sLongName      : T_MaxString;
    sLabel_8Char   : STRING(8);
    nObjectID      : UDINT;
    bState         : BOOL;
    bOnPressed     : BOOL;
END_STRUCT
END_TYPE

```

Name	Typ	Beschreibung
sName	STRING(32)	Eindeutiger Name der LogicSource
sLongName	T_MaxString	Name der LogicSource wie in der Artist Matrix definiert.
sLabel_8Char	STRING(8)	8 Zeichen Tastenbeschriftung wie in der Artist Matrix definiert.
nObjectID	UDINT	Eindeutige Objektidentifikationsnummer
bState	BOOL	Der aktuelle Status, wie er von RRCS empfangen wurde.
bOnPressed	BOOL	Für die zukünftige Nutzung.

5.2.2 E_RRCS_ErrorCodes

```

{attribute 'qualified_only'}
TYPE E_RRCS_ErrorCodes :
(
    Success                                     := 0,
    InternalApplicationError_TransactionKeyInvalid := 1,
    ConfigurationUnavailableNotYetLoaded_NetAddressInvalid := 2,
    NodeAddressInvalid_NodeAddressInvalid      := 3,
    PortAddressInvalid                          := 4,
    SlotNumberInvalid                          := 5,
    InputGainInvalid                           := 6,
    IP_addressInvalid                           := 7,
    TCP_PortInvalid                            := 8,
    LabelInvalid                               := 9,
    ConferencePositionInvalid                  := 10,
    OperationFailed_Artist_NetworkNotConnected := 11,
    OperationFailed_RouteNotExists             := 12,
    OperationImpossible_GatewayIsStandby       := 13,
    XML_RPC_RequestParametersWrong             := 14,
    Invalid_conference_or_conference_not_found := 15,
    Invalid_conference_member_or_not_found     := 16,
    Invalid_Priority                           := 17,
    Invalid_GPIO_Number                        := 18,
    Invalid_gain_value                         := 19,
    Timeout                                    := 20,
    No_permission                              := 21,
    ObjectNotExists                            := 22,
    No_USB_dongle                              := 23,
    Port_not_online                            := 24,
    Object_property_not_supported               := 25,
    Limit_exceeded                             := 26,
    Generic_error                              := 99,
    XML_Member_required_mandatory               := 101,
    XML_Member_Ambiguous                       := 102,
    XML_Member_Invalid                         := 103,
    XML_Member_Unknown                         := 104,
    XML_Member_Value_Type_boolean_required     := 105,
    XML_Member_Value_Type_int_required         := 106,
    XML_Member_Value_Type_struct_required      := 107,
    XML_Member_Value_Type_array_required       := 108,
    XML_Member_Value_Type_string_required      := 109,
    XML_Member_Value_invalid                   := 110,
    XML_Member_Value_out_of_range              := 111,
    Instruction_unsupported_with_current_configuration := 201,
    Duplicate_Configuration_Object             := 202,
    Addressed_Configuration_Object_invalid     := 203,
    Addressed_Configuration_Object_unsupported := 204,
    Address_already_in_use_in_configuration    := 205,

```

```
Configuration_Member_unsupported      := 206,  
Configuration_Member_ambiguous        := 207,  
Configuration_Value_invalid           := 208,  
Configuration_Value_already_inUse     := 209,  
Generic                              := 401,  
Unknown                              := 300000,  
NoErrCodeAvailable                   := 300001  
) DINT;  
END_TYPE
```

6 Beispiele

Für die Nutzung der RRCS Library stehen hier zwei aufeinander aufbauende Beispiele zur Verfügung:

- Tc3_RRCS_Sample01 dient zum Test der Kommunikation zwischen dem RRCS-Server und TwinCAT.
- Tc3_RRCS_Sample02: in diesem Beispiel wird der Zustand existierender 'Logic Sources' abgefragt. Auch kann der Zustand dieser Elemente geändert werden. Der Test der Kommunikation zwischen dem RRCS-Server und TwinCAT ist in diesem Beispiel inkludiert.

6.1 Tc3_RRCS_Sample01

Dieses einfache Beispiel dient ausschließlich dazu, die Kommunikation zwischen TwinCAT und dem RRCS-Server zu prüfen. Laden Sie das Beispiel hier herunter:

https://infosys.beckhoff.com/content/1031/tf6760_rrcs/Resources/19302661515.zip

In der Variablendeklaration (MAIN-Programm) werden alle benötigten Variablen deklariert und die Adressdaten analog zur Einstellung des RRCS-Servers vorgenommen. In diesem Fall:

- läuft der RRCS-Server auf einem Beckhoff IPC mit der IP-Adresse 192.168.42.59
- horcht der RRCS-Server auf Port 8193
- läuft das TwinCAT-Sample auf einem Beckhoff-Embedded-PC mit der IP-Adresse 192.168.42.3

RRCS 8.5.RR1-23fb8b4#10		
Disconnect-from-Artist Options...		
Time	Source	Message
13:19:06:498	XML-RPC-Server	Info: 'Server created.' Server-Name='RRCS Server (PID=5480)'
13:19:06:500	XML-RPC-Server	Info: 'Listener started on port 8193.' Server-IP='0.0.0.0:8193' Server-Name='RRCS Server (PID=5480)'
13:19:06:500	Online Manager	Network started.
13:19:14:813	Alarm Manager	Alarm activated, creation time="2025.05.09 13:19:07.153",update time="2025.05.09 13:19:07.153",path="Director",type="Autosave",severity
13:19:39:687	XML-RPC-Server	Info: 'Connection to client created.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50282' Server-Name='RRCS Server (PID=5480)'
13:19:39:689	XML-RPC-Server	Info: 'GetState(B0000000001) succeeded.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50282' Host='192.168.42.59:8193' User-Ag
13:19:39:722	XML-RPC-Server	Info: 'GetAlive(B0000000002) succeeded.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50282' Host='192.168.42.59:8193' User-Ag
13:19:39:752	XML-RPC-Server	Info: 'IsConnectedToArtist(B0000000003) succeeded: IsConnected=true' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50282' Host-
13:19:39:831	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000004) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
13:19:39:892	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000005) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
13:19:39:951	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000006) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
13:19:40:011	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000007) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
13:19:40:071	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000008) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
13:19:40:131	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B0000000009) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:50
Gateway is working (TCP-port 8193) Connected to Artist: 192.168.42.200		

Variablendeklaration:

```
PROGRAM MAIN
VAR
  (* the state of the statemachine *)
  eState: E_RRCS_State := IDLE;
  (* the IP-Address and port of the PC which runs the RRCS-      Server *)
  fbRRCScom: FB_RRCScom(
    ipAddr_Server:= '192.168.42.59',
    ipPort_Server:=8193):= (eTraceLevel := TcEventSeverity.Verbose);
  (* Retry in case of errors *)
  tonRetry: TON := (pt := T#5S);

  {region 'Get States'}
  (* functionblock used to get the state of the RRCS-Server *)
  fbGetState: FB_GetState(pfbRRCScom := ADR(fbRRCScom));
  {endregion}
END_VAR
```

Danach wird das Programm basierend auf einer Statemachine erstellt:

```
CASE eState OF
  CHECK_GATEWAY:
    tonRetry(IN := FALSE);
    fbGetState(bExecute:= TRUE);
    IF fbGetState.bError THEN
      eState:= ERROR;
    ELSIF NOT fbGetState.bBusy THEN
```



```
    IF fbGetState.sGatewayState = 'Working' THEN
        eState := IDLE;
    ELSE
        eState := ERROR;
    END_IF
END_IF

ERROR:
    fbGetState(bExecute:= FALSE);
    tonRetry(In:= NOT tonRetry.Q);
    IF tonRetry.Q THEN
        eState:= CHECK_GATEWAY;
    END_IF

END_CASE

IF NOT tonRetry.Q THEN
    eState := SEL(fbRRCScom.eCommState = E_CommState.ERROR, eState, ERROR);
END_IF
```

Starten Sie das Programm, ist der Status der Statemachine (eState) IDLE. Um den Status des RRCS-Servers abzufragen, müssen Sie den Status von eState auf CHECK_GATEWAY setzen. Siehe hierzu: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/2531621771.html.

The screenshot displays the TwinCAT 3 IDE. The top menu bar includes options like Datei, Bearbeiten, Ansicht, Projekt, Erstellen, Debuggen, TwinCAT, TwinCAT HMI, TwinSAFE, and PLC. The toolbar shows various icons for file operations and development. The main window is titled 'Tc3_RRCS_Sample01 - TcXaeShell' and shows the 'MAIN [Online]' project. The 'Überwachungsliste 1' (Watch List 1) is active, displaying a table of variables and their current values.

Ausdruck	Datentyp	Wert	Vorbereiteter Wert
eState	E_RRCS_STATE	IDLE	
fbRRCScom	FB_RRCScom		
tonRetry	TON		
bExecuteGetState	BOOL	FALSE	
fbGetState	FB_GetState		
i	INT	0	
ipResultMessage	I_TcMessage	16#FFFF838145...	
bExecute	BOOL	FALSE	
eRRCS_ErrorCode	E_RRCS_ERRORCO...	Success	
bBusy	BOOL	FALSE	
bError	BOOL	FALSE	
nStatusCode	UINT	200	
sGatewayState	STRING(32)	'Working'	

Below the table, the ladder logic code for the 'MAIN' project is visible. It shows a 'CASE' statement for 'eState' with 'IDLE' as the first case. The code includes calls to 'CHECK_GATEWAY', 'tonRetry', 'fbGetState', and 'bExecuteGetState'. It also includes conditional logic for 'bError' and 'bBusy' to set 'eState' to 'ERROR' or 'IDLE'.

```

1 CASE eState IDLE OF
2   CHECK_GATEWAY 1 :
3     tonRetry(IN FALSE := FALSE);
4     fbGetState(bExecute FALSE := bExecuteGetState FALSE);
5     bExecuteGetState FALSE := FALSE;
6     IF fbGetState.bError FALSE THEN
7       eState IDLE := ERROR 8;
8     ELSIF NOT fbGetState.bBusy FALSE THEN
9       IF fbGetState.sGatewayState 'Working' = 'Working' THEN

```

War die Anfrage erfolgreich, wechselt der Status des eState nach einigen Zyklen zurück auf IDLE und die Variable sGatewayState erhält den Wert 'Working'.

Trat bei der Abfrage ein Fehler auf, wechselt der Status des eState auf ERROR und es folgen weitere Versuche entsprechend jener Zeit, die in tonRetry angegeben wurde.

6.2 Tc3_RRCS_Sample02

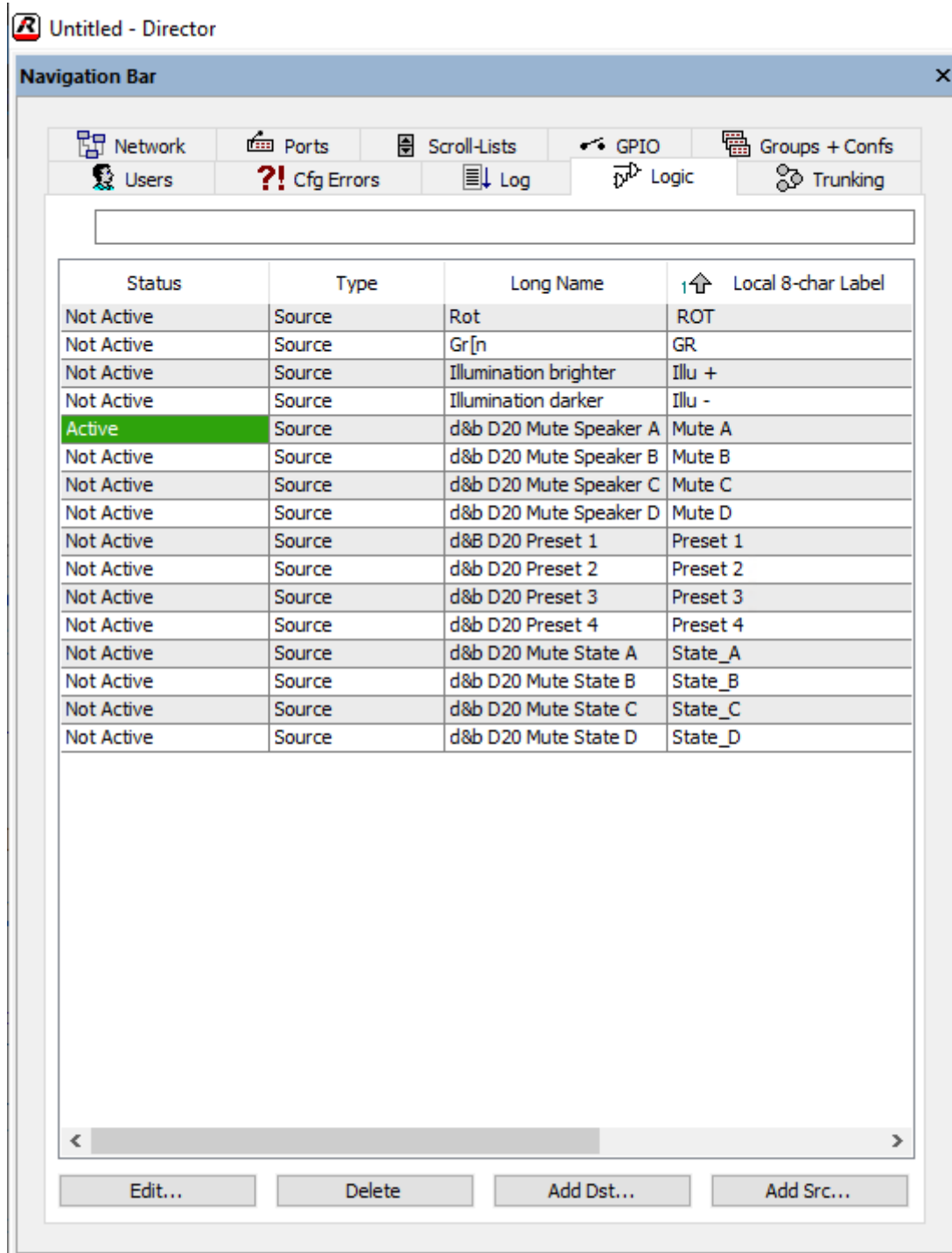
Wurde der GetState-Request aus dem vorherigen Beispiel erfolgreich beantwortet, kann die Funktionalität wie folgt erweitert werden:

- Eine Abfrage mittels GetAlive (Diese Funktion prüft, ob die Verbindung besteht.)
- Die Abfrage IsConnectedToArtist (Hiermit wird ermittelt, ob der RRCS-Server mit einer Artist-Matrix verbunden ist.)
- Das zyklische Abfragen der Zustände der 'Logic Sources'.
- Das Setzen der 'Logic Sources'.

Laden Sie das Beispiel hier herunter:

https://infosys.beckhoff.com/content/1031/tf6760_rrcs/Resources/19302663179.zip

Für dieses Beispiel wurden folgende 'Logic Sources' erstellt:



Basierend auf dem vorherigen Beispiel deklarieren Sie weitere Variablen:

```
PROGRAM MAIN
VAR
  (* the state of the statemachine *)
  eState: E_RRCS_State := IDLE;
  (* the IP-Address and port of the PC which runs the RRCS-Server *)
  fbRRCScom: FB_RRCScom(
    ipAddr_Server:= '192.168.42.59',
    ipPort_Server:=8193):= (eTraceLevel := TcEventSeverity.Verbose);
  (* Retry in case of errors *)
  tonRetry: TON := (pt := T#5S);
  tonWait: TON := (pt := T#70MS);
```

```

ui: UINT;

{region 'Get States'}
(* functionblock used to get the state of the RRCS-Server *)
fbGetState: FB_GetState(pfbRRCScom := ADR(fbRRCScom));
{endregion}

{region 'Get Alive'}
bExecuteGetAlive: BOOL := TRUE;
fbGetAlive: FB_GetAlive(pfbRRCScom := ADR(fbRRCScom));
{endregion}

{region 'Is connected to Artist'}
bExecuteGetConnectedToArtist: BOOL := TRUE;
fbIsConnectedToArtist: FB_IsConnectedToArtist(pfbRRCScom := ADR(fbRRCScom));
{endregion}

{region 'Get Logic Sources'}
bExecuteGetAllLogicSources: BOOL;
fbGetAllLogicSources_V2: FB_GetAllLogicSources_v2(pfbRRCScom := ADR(fbRRCScom));
tonGetLogicSourceStates: TON;
{endregion}

{region 'Set a Logic Source'}
bState: BOOL; // the required status
_bState: BOOL;
fbSetLogicSourceState: FB_SetLogicSourceState(pfbRRCScom := ADR(fbRRCScom));
{endregion}
bErrorObjId: BOOL;
END_VAR

```

Der Programmablauf aus dem vorherigen Sample wurde hier in der Statemachine um weitere Zustände ergänzt:

```

CASE eState OF
  CHECK_GATEWAY:
    tonRetry(IN := FALSE);
    fbGetState(bExecute:= bExecuteGetState);
    bExecuteGetState:= FALSE;
    IF fbGetState.bError THEN
      eState:= ERROR;
    ELSIF NOT fbGetState.bBusy THEN
      IF fbGetState.sGatewayState = 'Working' THEN
        eState := CHECK_ALIVE;
      ELSE
        eState := ERROR;
      END_IF
    END_IF

  CHECK_ALIVE:
    fbGetAlive(bExecute:= bExecuteGetAlive);
    bExecuteGetAlive:= FALSE;
    IF fbGetAlive.bError THEN
      eState:= ERROR;
    ELSIF NOT fbGetAlive.bBusy THEN
      IF fbGetAlive.bAlive THEN
        bExecuteGetConnectedToArtist:= TRUE;
        eState := CHECK_CONN_ARTIST;
      ELSE
        eState := ERROR;
      END_IF
    END_IF

  CHECK_CONN_ARTIST:
    fbIsConnectedToArtist(bExecute:= bExecuteGetConnectedToArtist);
    bExecuteGetConnectedToArtist:= FALSE;
    IF fbIsConnectedToArtist.bError THEN
      eState:= ERROR;
    ELSIF NOT fbIsConnectedToArtist.bBusy THEN
      IF fbIsConnectedToArtist.bIsConnected THEN
        eState := GET_SOURCES;
      ELSE
        eState := ERROR;
      END_IF
    END_IF

  GET_SOURCES:
    IF bState <> _bState THEN
      _bState := bState;
    END_IF

```

```

        eState := SET_SOURCE;
    ELSE
        tonGetLogicSourceStates(in := TRUE, pt:= T#50MS);
        IF tonGetLogicSourceStates.Q THEN
            tonGetLogicSourceStates(In := FALSE);
            bExecuteGetAllLogicSources := TRUE;
        END_IF
        fbGetAllLogicSources_V2(bExecute:= bExecuteGetAllLogicSources);
        bExecuteGetAllLogicSources := FALSE;
        IF fbGetAllLogicSources_V2.bError THEN
            fbGetAllLogicSources_V2(bExecute:= FALSE);
            eState:= ERROR;
        ELSIF NOT fbGetAllLogicSources_V2.bBusy AND bExecuteGetAllLogicSources THEN
            fbGetAllLogicSources_V2(bExecute:= FALSE);
            IF fbGetAllLogicSources_V2.eRRCS_ErrorCode <> E_RRCS_ErrorCodes.Success THEN
                eState:= ERROR;
            END_IF
        END_IF
    END_IF

SET_SOURCE:
    FOR ui := 0 TO PL_RRCS.nMaxLogicSources - 1 DO
        IF fbGetAllLogicSources_V2.aLogicSources[ui].sLabel_8Char = 'Mute A' THEN
            EXIT;
        END_IF
    END_FOR
    IF ui < PL_RRCS.nMaxLogicSources THEN // Logic Source found ....
        fbSetLogicSourceState(bExecute := TRUE, bState := bState,
nObjectID := fbGetAllLogicSources_V2.aLogicSources[ui].nObjectID,
        aLogicSources := fbGetAllLogicSources_V2.aLogicSources);
        IF fbSetLogicSourceState.bError THEN
            fbSetLogicSourceState(bExecute := FALSE, aLogicSources :=
fbGetAllLogicSources_V2.aLogicSources);
            eState:= ERROR;
        ELSIF NOT fbSetLogicSourceState.bBusy THEN
            fbSetLogicSourceState(bExecute := FALSE, aLogicSources :=
fbGetAllLogicSources_V2.aLogicSources);
            eState := WAIT;
        END_IF
    END_IF

WAIT:
    tonWAIT(In := NOT tonWAIT.Q);
    IF tonWAIT.Q THEN
        tonWAIT(In := FALSE);
        eState := GET_SOURCES;
    END_IF

ERROR:
    fbGetState(bExecute:= FALSE);
    tonRetry(In:= NOT tonRetry.Q);
    IF tonRetry.Q THEN
        bExecuteGetState:= TRUE;
        eState:= CHECK_GATEWAY;
    END_IF

END_CASE

IF NOT tonRetry.Q THEN
    eState := SEL(fbRRCScom.eCommState = E_CommState.ERROR, eState, ERROR);
END_IF

```

Die Anzahl und der Zustand der 'Logic Sources' wird pollend abgefragt. Die gefundenen werden wie folgt in Arrayform zur Verfügung gestellt:

Ausdruck	Applikation	Datentyp	Wert
MAIN.fbGetAllLogicSources_v2.nLogicSourceCount	Tc3_RRCS_Sample0...	UINT	16
MAIN.fbGetAllLogicSources_v2.aLogicSources	Tc3_RRCS_Sample0...	ARRAY [0...(PL_...	
aLogicSources[0]		ST_LogicSource	
sName		STRING(32)	'LogicSource#1'
sLongName		T_MaxString	'd&b D20 Preset 2'
sLabel_8Char		STRING(8)	'Preset 2'
nObjectID		UDINT	349587280
bState		BOOL	FALSE
bOnPressed		BOOL	FALSE
aLogicSources[1]		ST_LogicSource	
aLogicSources[2]		ST_LogicSource	
aLogicSources[3]		ST_LogicSource	
aLogicSources[4]		ST_LogicSource	
aLogicSources[5]		ST_LogicSource	
aLogicSources[6]		ST_LogicSource	
aLogicSources[7]		ST_LogicSource	
aLogicSources[8]		ST_LogicSource	
aLogicSources[9]		ST_LogicSource	
aLogicSources[10]		ST_LogicSource	
aLogicSources[11]		ST_LogicSource	
sName		STRING(32)	'LogicSource#5'
sLongName		T_MaxString	'd&b D20 Mute Spea...
sLabel_8Char		STRING(8)	'Mute A'
nObjectID		UDINT	669062033
bState		BOOL	TRUE
bOnPressed		BOOL	FALSE
aLogicSources[12]		ST_LogicSource	
sName		STRING(32)	'LogicSource#6'
sLongName		T_MaxString	'Illumination darker'
sLabel_8Char		STRING(8)	'Illu -'
nObjectID		UDINT	83566801
bState		BOOL	FALSE
bOnPressed		BOOL	FALSE
aLogicSources[13]		ST_LogicSource	
aLogicSources[14]		ST_LogicSource	
aLogicSources[15]		ST_LogicSource	
aLogicSources[16]		ST_LogicSource	

Jede der 'Logic Sources' ist mittels eindeutiger ObjektID ansprechbar.

Wird der Zustand der Variable `bState` geändert, wird versucht, den Zustand dieser 'Logic Source' auf den gewünschten Wert zu setzen.

Dies geschieht mit einer Instanz des Funktionsbausteins `FB_SetLogicSource`.

In der GUI des RRCS-Servers wird das Setzen des Wertes der LogicSource#5 wie folgt dargestellt:

RRCS 8.5.RR1-23fb8b4#10		
Disconnect-from-Artist Options...		
Time	Source	Message
13:10:39:063	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B00000000687) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
13:10:39:132	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B00000000688) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
13:10:39:182	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B00000000689) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
13:10:39:242	XML-RPC-Server	Info: 'GetAllLogicSources_v2(B00000000690) succeeded: Returning 16 logic sources.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
13:11:02:853	XML-RPC-Server	Info: 'Initiating SetLogicSourceState(B00000000691, {obj-ID}669062033, true).' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
13:11:02:927	XML-RPC-Server	Info: 'SetLogicSourceState(B00000000691, {obj-ID}669062033, true) succeeded.' Server-IP='192.168.42.59:8193' Client-IP='192.168.42.3:49926' Host=...
<		
Gateway is working (TCP-port 8193) Connected to Artist: 192.168.42.200		

7 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline: +49 5246 963-157

E-Mail: support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline: +49 5246 963-460

E-Mail: service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon: +49 5246 963-0

E-Mail: info@beckhoff.com

Internet: www.beckhoff.com

Trademark statements

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mehr Informationen:

www.beckhoff.com/entertainment-industrie

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com

