BECKHOFF New Automation Technology

Manual | EN

TwinCAT 3

Riedel Communications | RRCS

2025-07-23 | Version: 1.2.0

Table of Contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

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.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/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.

3 Overview

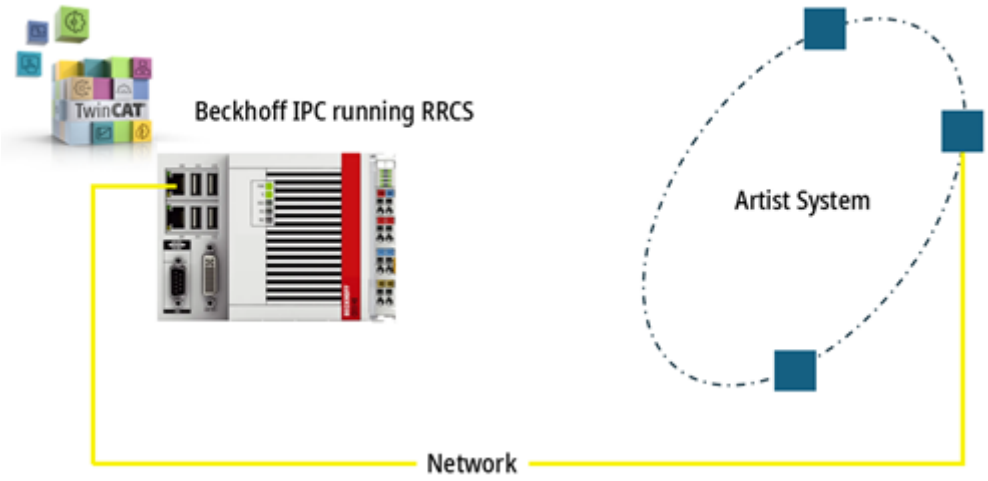
This example describes the communication possibilities between Beckhoff and Riedel Communications.

Riedel Communications

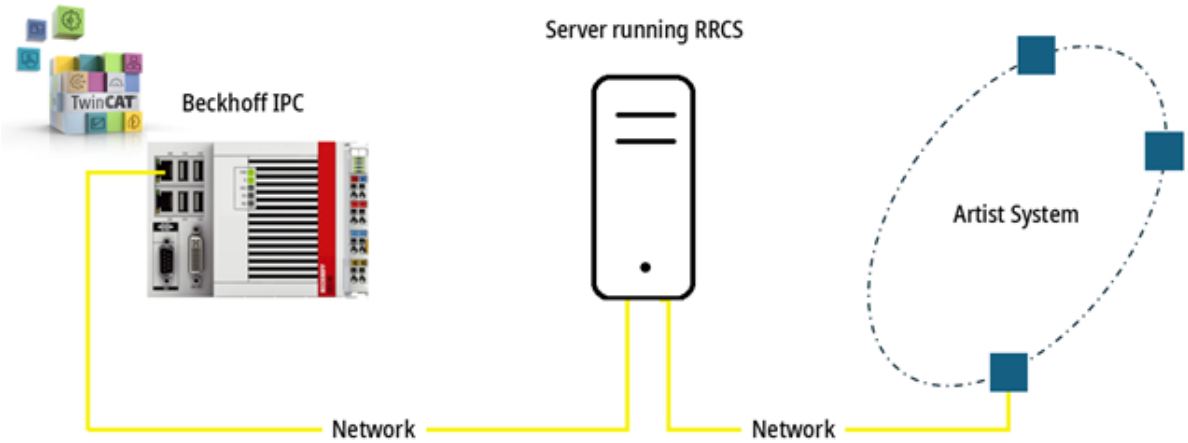
Riedel Communications GmbH & Co. KG develops, manufactures and distributes intercom, fiber optic, radio and audio network systems for use in broadcasting, events, theater and industry.

With the Riedel Router Control Software (RRCS), Riedel offers a universal, XML-based interface that can be used to control Artist series intercom systems.
To use RRCS, you need to install the RRCS server. This server runs on PCs with the Windows operating system. There are two options:

- Installation of the RRCS server on Beckhoff IPCs on which the Beckhoff TwinCAT program is also running



- Installation of the RRCS server on any server



Beckhoff

Beckhoff provides a library with which Beckhoff control systems can communicate bidirectionally with Riedel Artist systems via RRCS.
This library is described below and is part of the samples. You can download the samples in the chapter Samples [▶ 21]. To use the library, you need a TC1200 license and a license for TF6760. The library and the samples are subject to the disclaimer [▶ 8] in chapter 2. In addition, two exemplary implementations are listed in this documentation.

Summary and benefits

- Combines Beckhoff's open PC and EtherCAT-based control technology with Riedel communication systems.
- Direct support of the RRCS protocol from Riedel in TwinCAT
- Enables communication with Riedel Artist and Bolero systems as well as the use of Riedel SmartPanels.
- Facilitates dynamic control of audio circuits and integration with theater and show announcement systems.
- System integrators can now integrate open PC and EtherCAT-based control technology components from Beckhoff into the Riedel system, and vice versa.
- Beckhoff supports the essential parts of the RRCS protocol from Riedel in TwinCAT on the basis of TwinCAT 3 IoT HTTPS/REST (TF6760).

- This enables communication with Riedel Artist and Bolero systems and the implementation of Riedel SmartPanels.
- Based on TwinCAT Speech (TF4500) error messages or any type of announcements can be created by using our text to speech function blocks. Also, audio files can be played.
- The audio data created by TwinCAT Speech (or other tools) can be send via DANTE to the Riedel System by using the DANTE virtual audio card on Beckhoff Windows 10 or 11 based systems.
- In a theatre context, possible use cases can be the dynamic change and switching of audio circuits based on logic and priorities running in the Beckhoff control system or, in combination with theater stage management, the show control including cue-light signaling and announcements.

The control elements for the end user can be selected and combined, from the following:

- Beckhoff multi-touch Control Panels and TwinCAT HMI
- In combination with classic light switches or control wheels and other mechanical inputs
- Riedel SmartPanels or wireless Bolero beltpacks as control and monitoring interfaces

As the Beckhoff control system offers almost endless communication and control possibilities, controlling audio/video systems, stage and show applications, and even building automation with temperature control, along with blinds and general lights, is possible.

3.1 System Requirements

The following requirements must be met to execute the sample projects:

Technical data	Description
TwinCAT version	TwinCAT 3.1 Build 4022.20 or higher
Required TwinCAT licenses	TC1200 | TwinCAT 3 PLC
	TF6760 | TwinCAT 3 IoT HTTPS/REST
Riedel Communcations RRCS server software	Further information: RRCS

Test network

Before launching a project that involves the use of any network protocol, it is crucial to ensure that the network being used is functioning correctly. We recommend starting with a simple test where the Beckhoff device has a direct connection to the device it should communicate with. With no Firewalls or similar active.

1. First, verify the hardware components by checking cable connections and ensuring that routers or switches are powered on and functional.
2. Next, inspect the network settings such as IP addresses, subnet masks, and firewall settings on both the Beckhoff device and the device it is intended to communicate with.

⇨ After that basic test works, you can configure the firewall as described here: https://download.beckhoff.com/download/document/ipc/industrial-pc/ipc_security_guideline_en.pdf

3.2 Revision control

Date	Version	Change
2025-05-08	3.0.0.0	First version

4 Installation

Installation of the Riedel Communications RRCS software

Depending on the selected setup, install the software on the corresponding device. (See Overview [▶ 9]) The procedure is described in the RRCS - Quick Startup Guide from Riedel.

Installing TwinCAT

The Beckhoff Industrial PC must have the runtime environment (XAR) installed. Details can be found here:
https://infosys.beckhoff.com/index.php?content=../content/1033/tc3_installation/179470219.html

You must also install the TwinCAT Function TF6760. Details can be found here:
https://infosys.beckhoff.com/index.php?content=../content/1033/tf6710_tc3_iot_functions/2544553995.html

5 Programming

Bidirectional communication between TwinCAT and the Riedel Communications RRCS server takes place using the function blocks described below, which are contained in the Tc3_RRCS library.

To be able to use function blocks of the Tc3_RRCS library, you must install and license the TwinCAT 3 Function TF6760 IoT HTTPS/REST
!

5.1 Function blocks

5.1.1 FB_RRCScom

This function block is required for communication with the RRCS server

This function block should only be instantiated and not called implicitly!
The call is made using pointers from the function blocks described below, e.g. from FB_GetState.

The IP address and port of the RRCS server must be specified in the variable declaration:

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

Outputs

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

Name	Type	Description
eCommState	E_CommState	The state of the function block.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status
sResponse	STRING(..)	String with the unprocessed response from the RRCS server. The maximum length of the string is set in the parameter list of the library.
_pfbSB	POINTER	Only for internal use

5.1.2 FB_GetState

This function block queries the state of the RRCS server.

Inputs

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

Name	Type	Description
bExecute	BOOL	The function block is enabled by a positive edge at this input.

Outputs

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

Name	Type	Description
eRRCS_ErrorCode	E_RRCS_ErrorCodes [▶ 19]	Error code received from RRCS_Servers.
bBusy	BOOL	Is TRUE as long as the function block is busy with the query and no error occurs.
bError	BOOL	Is TRUE if an error occurred during the query.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status
sGatewayState	STRING(32)	Information from the RRCS server, possible states: • Working • Standby

FB_GetState has an extended FB_init method in which an instance of FB_RRCScom must be referenced using a pointer:

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

5.1.3 FB_GetAlive

To ensure that there is a correct connection between TwinCAT and the RRCS server, the function block FB_GetAlive can be used.

Inputs

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

Name	Type	Description
bExecute	BOOL	The function block is enabled by a positive edge at this input.

Outputs

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

Name	Type	Description
eRRCS_ErrorCode	E_RRCS_ErrorCodes [▶ 19]	Error code received from RRCS_Servers.
bBusy	BOOL	Is TRUE as long as the function block is busy with the query and no error occurs.
bError	BOOL	Is TRUE if an error occurred during the query.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status
bAlive	BOOL	Is TRUE if communication to the RRCS server is working.

FB_GetAlive has an extended FB_init method in which an instance of FB_RRCScom must be referenced using a pointer:

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

5.1.4 FB_IsConnectedToArtist

This function block is used to query whether the RRCS server is communicating with an Artist device.

Inputs

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

Name	Type	Description
bExecute	BOOL	The function block is enabled by a positive edge at this input.

Outputs

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

Name	Type	Description
eRRCS_ErrorCode	E_RRCS_ErrorCodes [▶ 19]	Error code received from RRCS_Servers.
bBusy	BOOL	Is TRUE as long as the function block is busy with the query and no error occurs.
bError	BOOL	Is TRUE if an error occurred during the query.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status
bIsConnected	BOOL	Is TRUE if the RRCS server is communicating with an Artist device.

FB_IsConnectedToArtist has an extended FB_init method in which an instance of FB_RRCScom must be referenced using a pointer:

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

5.1.5 FB_GetAllLogicSources_V2

This function block queries the 'Logic sources' defined in the Artist device.

Inputs

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

Name	Type	Description
bExecute	BOOL	The function block is enabled by a positive edge at this input.

Outputs

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

Name	Type	Description
eRRCS_ErrorCode	E_RRCS_ErrorCodes [▶ 19]	Error code received from RRCS_Servers.
bBusy	BOOL	Is TRUE as long as the function block is busy with the query and no error occurs.
bError	BOOL	Is TRUE if an error occurred during the query.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status
nLogicSourceCount	UINT	The number of 'Logic Sources' defined in the Artist device.
aLogicSources	ARRAY[ ] OF ST_LogicSource [▶ 19]	Listing of all defined 'Log Sources' in the form of an array. The maximum number of array entries is defined in the parameter list.

FB_GetAllLogicSources_V2 has an extended FB_init method in which an instance of FB_RRCScom must be referenced using a pointer:

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

5.1.6 FB_SetLogicSourceState

This function block offers the option of changing the state of a 'Logic Source'. This is done using the unique ObjectID.

Inputs

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

Name	Type	Description
bExecute	BOOL	The function block is enabled by a positive edge at this input.
nObjectID	UDINT	Logic Sources' are accessed via the unique ObjectID.
bState	BOOL	Specification of the desired state.
aLogicSources	ARRAY[ ] OF ST_LogicSource [▶ 19]	Listing of all defined 'Log Sources' in the form of an array. The maximum number of array entries is defined in the parameter list.

Outputs

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

Name	Type	Description
eRRCS_ErrorCode	E_RRCS_ErrorCodes [▶ 19]	Error code received from RRCS_Servers.
bBusy	BOOL	Is TRUE as long as the function block is busy with the query and no error occurs.
bError	BOOL	Is TRUE if an error occurred during the query.
nStatusCode	UINT	Servers deliver the status code as a response to every HTTP request. The server uses this code to inform the client whether the request was successful. Normally 200 (OK) is returned. Detailed information on this: developer.mozilla.org/en/docs/Web/HTTP/Reference/Status

FB_SetLogicSourceState has an extended FB_init method in which an instance of FB_RRCScom must be referenced using a pointer:

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

5.2 Data types

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	Type	Description
sName	STRING(32)	Unique name of the LogicSource
sLongName	T_MaxString	Name of the LogicSource as defined in the artist matrix.
sLabel_8Char	STRING(8)	8-character button labeling as defined in the artist matrix.
nObjectID	UDINT	Unique object identification number
bState	BOOL	The current status as received by RRCS.
bOnPressed	BOOL	For future use.

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 Samples

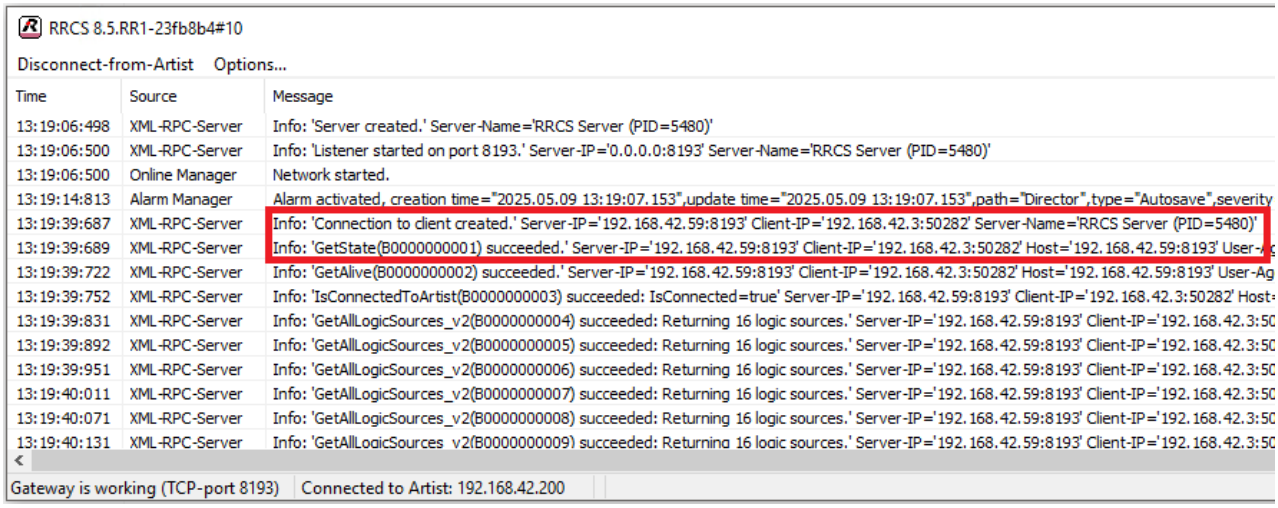
Two consecutive samples are available here for using the RRCS Library:

- Tc3_RRCS_Sample01 is used to test the communication between the RRCS server and TwinCAT.
- Tc3_RRCS_Sample02: in this sample, the state of existing 'Logic Sources' is queried.
  The state of these elements can also be changed. The test of the communication between the RRCS server and TwinCAT is included in this sample.

6.1 Tc3_RRCS_Sample01

This simple sample is used exclusively to check the communication between TwinCAT and the RRCS server. Download the example here:
https://infosys.beckhoff.com/content/1033/tf6760_rrcs/Resources/19302661515.zip
All the required variables are declared in the variable declaration (MAIN program) and the address data is set in the same way as for the RRCS server. In this case:

- the RRCS server runs on a Beckhoff IPC with the IP address 192.168.42.59
- the RRCS server listens on port 8193
- the TwinCAT sample runs on a Beckhoff Embedded PC with the IP address 192.168.42.3



Variable declaration:

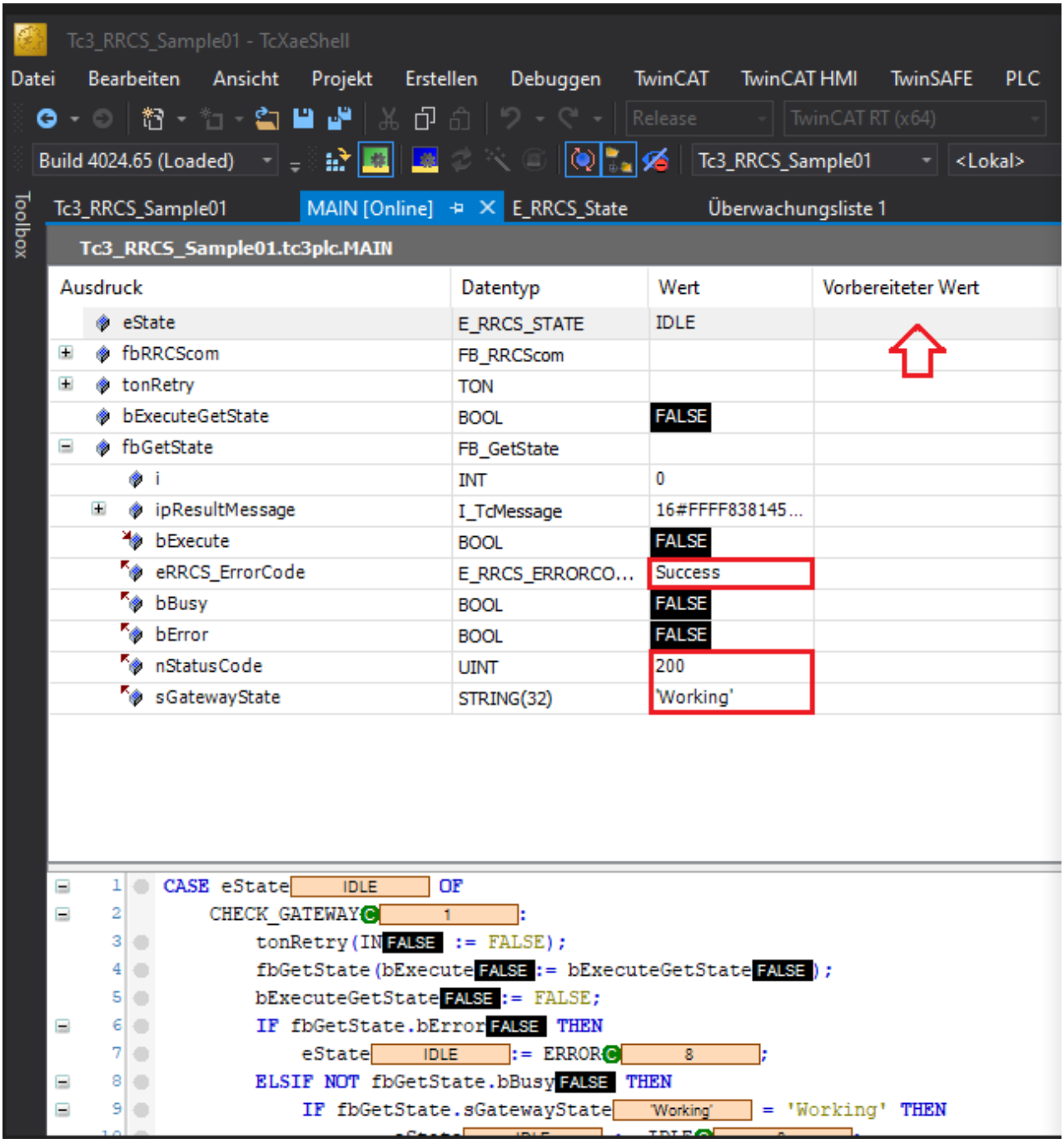
```
PROGRAM MAIN
VAR
    (* the state of the statemachine *)
    eState: E_RRCS_State := IDLE;
    (* the IP-Address and port of the PC which runs the RRCS-    Server *)
    fbRRCScom: FB_RRCScom(
        ipAddr_Server:= '192.168.42.59',
        ipPort_Server:=8193):= (eTraceLevel := TcEventSeverity.Verbose);
    (* 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
```

The program is then created based on a state machine:

```
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
```

If you start the program, the status of the state machine (eState) is IDLE. To query the state of the RRCS server, you must set the state of eState to CHECK_GATEWAY. See also: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/2531621771.html.



If the request was successful, the state of the eState changes back to IDLE after a few cycles and the variable `sGatewayState` receives the value 'Working'.
If an error occurs during the query, the state of the eState changes to ERROR and further attempts are made according to the time specified in `tonRetry`.

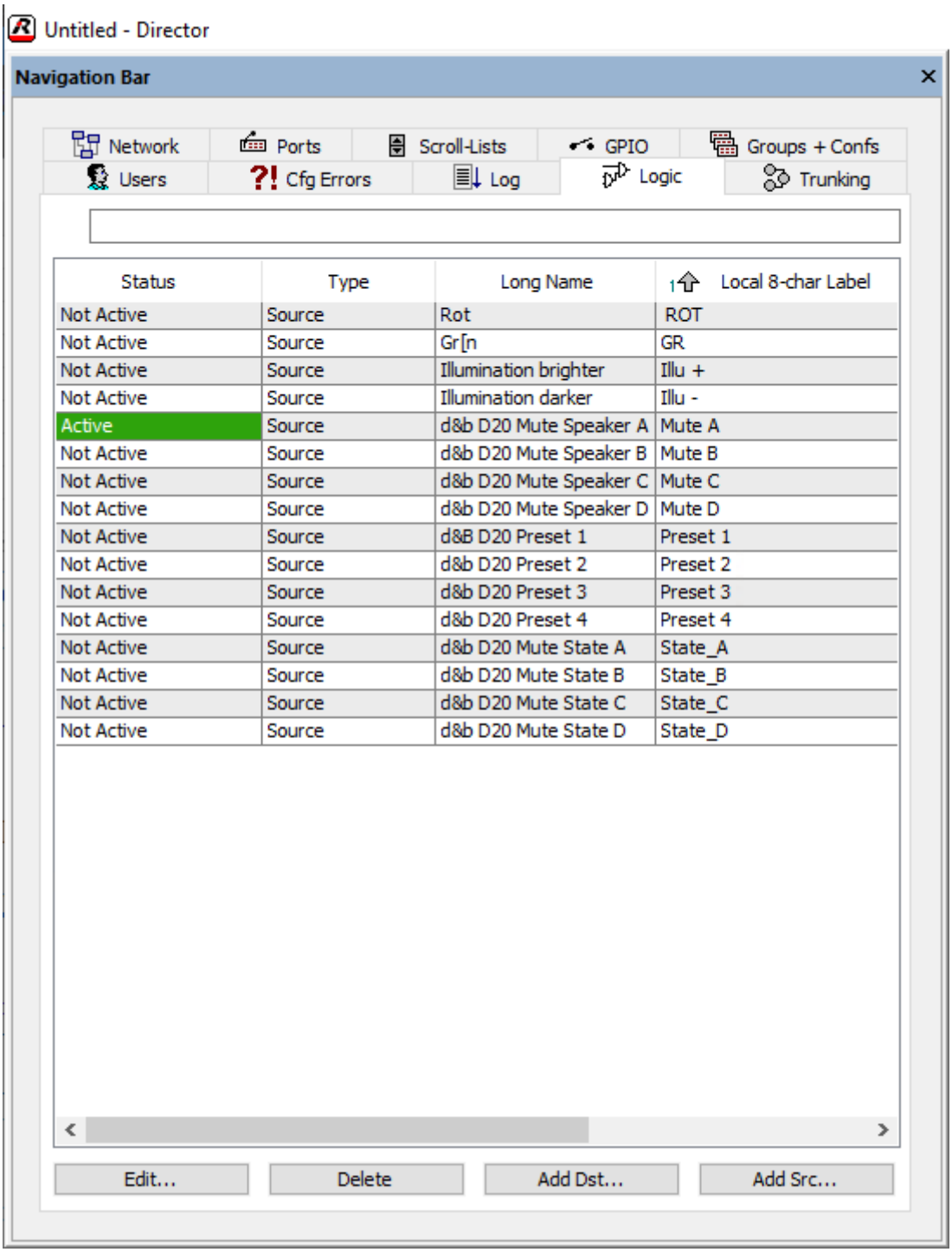
6.2 Tc3_RRCS_Sample02

If the GetState request from the previous sample was answered successfully, the functionality can be extended as follows:

- A query using `GetAlive` (This function checks whether the connection exists.)
- The query `IsConnectedToArtist` (This is used to determine whether the RRCS server is connected to an artist matrix)
- The cyclic querying of the states of the 'Logic Sources'.
- Setting the 'Logic Sources'.

Download the example here:
https://infosys.beckhoff.com/content/1033/tf6760_rrcs/Resources/19302663179.zip

The following 'Logic Sources' were created for this sample:



Status	Type	Long Name	Local 8-char Label
Not Active	Source	Rot	ROT
Not Active	Source	Gr[n	GR
Not Active	Source	Illumination brighter	Illu +
Not Active	Source	Illumination darker	Illu -
Active	Source	d&b D20 Mute Speaker A	Mute A
Not Active	Source	d&b D20 Mute Speaker B	Mute B
Not Active	Source	d&b D20 Mute Speaker C	Mute C
Not Active	Source	d&b D20 Mute Speaker D	Mute D
Not Active	Source	d&B D20 Preset 1	Preset 1
Not Active	Source	d&b D20 Preset 2	Preset 2
Not Active	Source	d&b D20 Preset 3	Preset 3
Not Active	Source	d&b D20 Preset 4	Preset 4
Not Active	Source	d&b D20 Mute State A	State_A
Not Active	Source	d&b D20 Mute State B	State_B
Not Active	Source	d&b D20 Mute State C	State_C
Not Active	Source	d&b D20 Mute State D	State_D

Based on the previous sample, declare further variables:

```
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
```

The program sequence from the previous sample was supplemented with additional states in the state machine:

```
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;
```

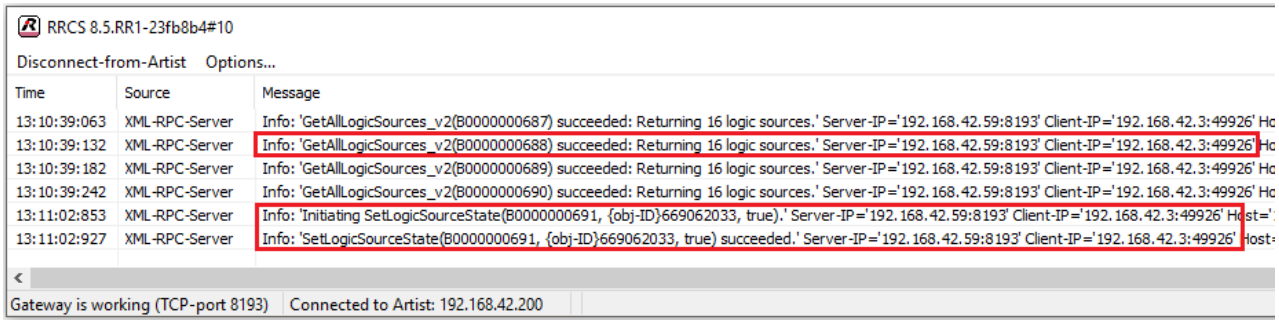
```
            eState := SET_SOURCE;
        ELSE
            tonGetLogicSourceStates(in := TRUE, pt:= T#50MS);
            IF tonGetLogicSourceStates.Q THEN
                tonGetLogicSourceStates(In := FALSE);
                bExecuteGetAllLogicSources := TRUE;
            END_IF
            fbGetAllLogicSources_V2(bExecute:= 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
```

The number and state of the logic sources is polled. Those found are made available in array form as follows:

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	

Each of the 'Logic Sources' can be addressed by means of a unique ObjectID.
If the state of the variable `bState` is changed, an attempt is made to set the state of this 'Logic Source' to the desired value.
This is done with an instance of the function block FB_SetLogicSource.

In the GUI of the RRCS server, the setting of the value of LogicSource#5 is displayed as follows:



7 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	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.

More Information:
www.beckhoff.com/entertainment-industry

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