BECKHOFF New Automation Technology

Manual | EN

TwinCAT 3

Q-Sys - QRC

2022-12-13 | Version: 1.3.2

Table of Contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared 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 prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

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

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

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 Overview

QSC is a professional audio/video system solutions provider. Its software-based platform is called Q-SYS. Q-SYS is designed to allow third-party systems to control and/or monitor various aspects of the system by writing your own code using different communication protocols.

The Q-SYS software supports following ways of external control:

- Named Controls – Controls that have been placed at Named Control pane. The names of the controls must be different. [This is part of the Q-SYS control level.]
- Component Control – Control all controls within any component by customizing the name of component to make it unique. [This is part of the Q-SYS component level.]
- Mixer Control – Specialized control of mixers using mixer concepts. [This is part of the Q-SYS component level.]

Basically, there are two different protocols provided by QSC to access the three above mentioned external controls for Q-SYS. They are called "Q-SYS External Control Protocol" and "**Q**-SYS **R**emote **C**ontrol" (QRC in the following).

- **Q-SYS External Control Protocol:**

  Q-SYS External Control Protocol is based on ASCII and using TCP/IP connection on port 1702 and it requires the use of Named Controls for any control which should be externally controlled. This means it only supports Q-SYS control level functions.

- **QRC:**

  QRC is the latest and most advanced protocol provided by QSC to allow an external control system (e.g. TwinCAT) to control various functions within Q-SYS. The QRC protocol is based on JSON-RPC version 2.0 and is using TCP/IP connection on port 1710. QRC supports the use of all three above mentioned controls: Named Controls, Component Control and Mixer Control. Based on that it allows the external access at control level and component level.

The precondition of external access at **control level**, is that every control in Q-SYS you want to be controlled, must be dragged into the Named Controls pane and the name of it must be unique.

In this document, how QRC can be used with Beckhoff controllers (TwinCAT software) will be explained. An example code called `Tc3_Qrc` library will also be provided in attachment.

The `Tc3_Qrc` library enables the implementation of one or more QRC external clients in the TwinCAT PLC. With its help, a Q-SYS Core can be controlled directly from a TwinCAT program.

QRC controls can be mapped to any data types in TwinCAT. This allows a large range of communication possibilities for the system integrator.

The QRC specification can be found here.

The QRC specification and its features are designed and developed by QSC, specification may be changed in the future.

Further information about the activities of Beckhoff in the market stage and show can be found on our website at: PC-based Control for Stage and Show Technology

System Requirement:

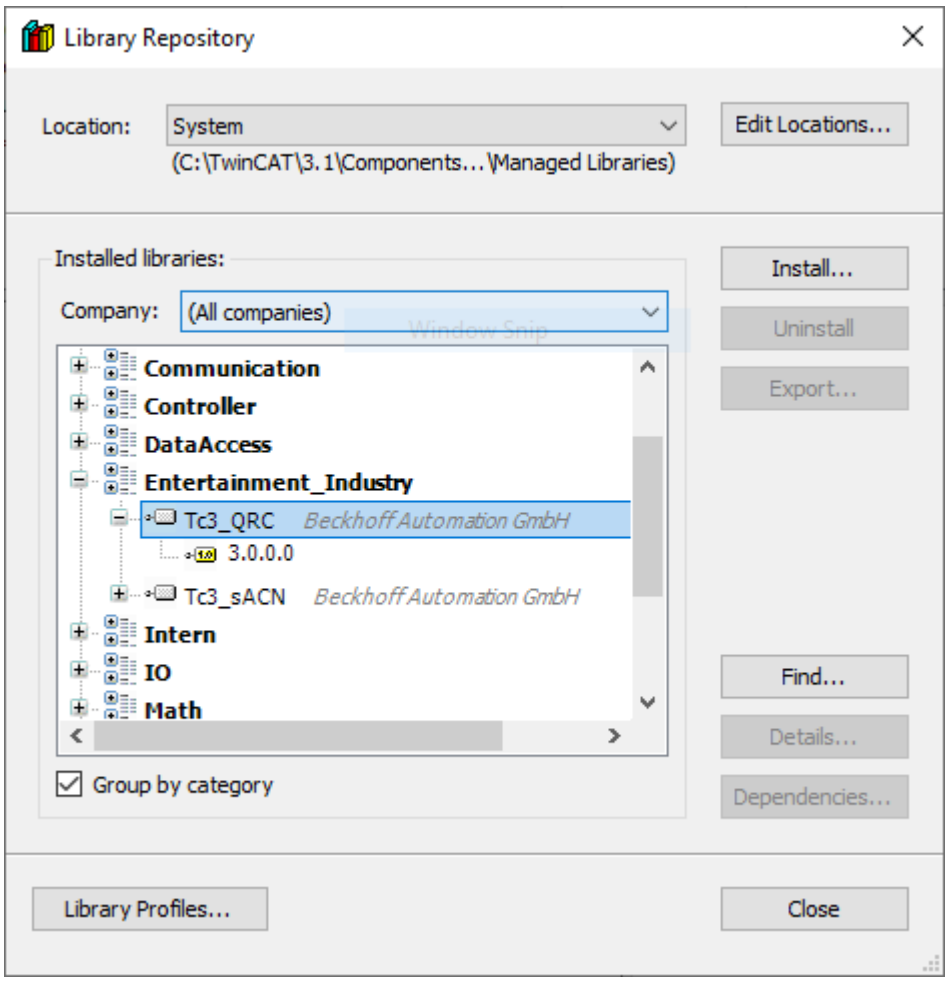
Technical Data	Requirement
TwinCAT version	TwinCAT 3.1 build 4022.20 or higher
Visual Studio version	Visual Studio 2013 or higher
Required TwinCAT license	TF6310 licence

2.1 Update History

[Version 3.0.0.0] – 2020.12.15

Changed:

- Changed the major version number of this library to 3.x.x.x because of TwinCAT 3.
- Move this library file into the library category “Entertainment_Industry”.



[Version 1.1.2.0] – 2020.11.12

Added:

- Added a method FB_exit [▶ 16] for online changing the input parameters of FB_init [▶ 14].

Changed:

- Bug fixed.

[Version 1.1.0.0] - 2020.03.10

Added:

- Added a new function block FB_QRC_Snapshot [▶ 38] for Snapshot Bank.
- Extended function block FB_QRC_ChangeGroup [▶ 28] with an additional method AddSnapshotControl [▶ 31] for adding snapshot component in a change group.
- Added Support about extraction frame of snapshot control. Read the section Workflow about extraction of snapshot properties [▶ 20] for more information.
- Extended function block FB_QRC_ResExtract [▶ 16] with an additional method Clear [▶ 18] for clearing internal storage.
- Added modifier for each method. (Internal methods can't be accessed anymore starting from this version.)
- Added a property `sTxFrame` to all QRC Command function block to read the QRC sending frame easily without the connection function block.

Changed:

- Adjusted the input variable of function block FB_QRC_LoopPlayer [▶ 37].
- Adjusted the severity of some events.

- Adjusted the variable name and type of structure ST_FileSpec [▶ 41] and ST_JobSpec [▶ 41] for better understanding.
- Adjusted the prefix of property name with the type ARRAY to fit TwinCAT 3 programming conventions.
- Bug fixed.

Removed:

- Removed the Get method from property sTxFrame of I_Connect [▶ 15].

3 Installation

The Q-SYS Core is considered as the server and the TwinCAT automation platform is considered as the client. The Q-SYS Core should load a Q-SYS design file and switch to Run to connect to TwinCAT automation platform. (In Q-SYS Designer this process is called Run mode).

Alternatively, if there is no Q-SYS hardware available, a Q-SYS design file can be simulated (in Q-SYS Designer the simulation process is called Emulate mode) on Q-SYS designer software without hardware. More information can be found at website Q-SYS help portal.

Before using this `tc3_Qrc` library, target controls and components must be set up in Q-SYS Designer:

- For target controls, they must be dragged to the Named Controls pane.
- For target components, their names must be customized and unique.
- For mixer Control and snapshot control, they are also types of component control and they should be prepared like target components.

In following paragraphs, the words **Q-SYS device** represents Q-SYS Core in Run mode or Q-SYS designer software in Emulate mode.

4 Programming

This sample project generally consists of three modules, an encode module, a communication module, and a decode module. Additionally, two interfaces are designed. An interface is used to enable the data exchange between encode module and communication module, and the other interface is used to enable the data exchange between communication module and decode module.

4.1 Function Blocks

In this project, all function blocks are mainly divided into 3 parts. The function block FB_Connect [▶ 13], which belongs to the communication module, is used for creating TCP connection; 7 function blocks [▶ 22], which belong to the encode module, are used to encode QRC frame. Furthermore, a helper function block FB_QRC_ResExtract [▶ 16], which belongs to decode module, is used for extract QRC response frame.

4.1.1 FB_Connect

This function block enables to establish or terminate a TCP connection.

Syntax

```
FUNCTION_BLOCK FB_Connect IMPLEMENTS I_Connect
VAR_OUTPUT
    bBusy                       : BOOL;
    bError                      : BOOL;
    eState                      : E_SocketConnectionState;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bBusy	BOOL	Is TRUE as long as the asynchronous request is still active. Is FALSE if the request was completed or an error occurs.
bBusy	BOOL	Is set if an error occurs during the execution of the function block. Error details are located in the "Error List" window.
eState	E_SocketConnectionState	Returns the current connection state. • eSOCKET_DISCONNECTED: disconnected • eSOCKET_SUSPENDED: state between connected and disconnected • eSOCKET_CONNECTED: connected
ipResult Message	I_TcMessage	Enables error handling with the `Tc3_EventLogger`.

Methods

Name	Description
FB_init	Initialization method
Connect	Establish a TCP connection.
Disconnect	Terminate a TCP connection.
Send	Send the QRC frame.
Receive	Receive the QRC frame.
FB_exit	Online Change method

Because all methods are asynchronous and they need more than one cycle to finish working, only one method could be invoked at the same time. Therefore, check the output parameter `bBusy` when one of these methods is being called.

Properties

Properties	Type	Access	Description
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Get	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the received QRC response frame can be get with this property.
sTxFrame	STRING(QRC_BUFFER_SIZE)	Set	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the QRC frame to be sent can be set with this property.

Interface

Name	Description
I_Connect	The interface that defines communication related methods.

4.1.1.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    sSrvNetID                   : T_AmsNetID := '';
    sRemoteHost                 : T_IPv4Addr := '127.0.0.1';
    tReconnect                  : TIME       := T#30s;
    iResExtract                 : I_ResExtract;
END_VAR
```

VAR_INPUT

sSrvNetID: AMS Net Id. For the local computer (default) an empty string may be specified.

sRemoteHost: Target IPv4 address.

tReconnect: Cooldown time for recreating a TCP connection after a TCP connection has been terminated.

iResExtract: The function block that implements the interface I_ResExtract [▶ 18].

Example:

Declaration of the function block `FB_Connect`:

```
PROGRAM MAIN
VAR
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbResExtract        : FB_QRC_ResExtract;
END_VAR
```

4.1.1.2 I_Connect

METHODS

Connect: Create a TCP connection.

Disconnect: Terminate a TCP connection.

Send: Send QRC frames.

M_Receive: Receive QRC frames.

PROPERTIES

Properties	Type	Access	Description
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Get	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the received QRC response frame can be queried with this property.
sTxFrame	STRING(QRC_BUFFER_SIZE)	Set	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the QRC frame to be sent can be set.

Connect

This method enables creating a TCP connection.

```
Method Connect              : BOOL
```

This process is finished as soon as the return value is TRUE.

Disonnect

This method enables terminating a TCP connection.

```
Method Disonnect            : BOOL
```

This process is finished as soon as the return value is TRUE.

Send

This method enables sending a QRC frame and to get the response frame from Q-SYS device automatically after sending.

```
Method Send                 : I_ResExtract
```

This method is finished as soon as the falling edge of `bBusy` occurs and property `aRxFrame` is not empty. The response frame can be fetched at property `aRxFrame`.

Receive

This method enables receiving a QRC frame.

```
Method Receive            : I_ResExtract
```

This method is finished as soon as the falling trigger of `bBusy` is triggered and property. `aRxFrame` is not empty. The response frame can be fetched at property `aRxFrame`.

aRxFrame

List of received QRC response frames.

```
PROPERTY aRxFrame       : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString
```

sTxFrame

A QRC frame which is ready to send to Q-SYS device.

```
PROPERTY sTxFrame         : STRING(QRC_BUFFER_SIZE)
```

4.1.1.3 FB_exit

Syntax

```
Method FB_exit : BOOL
```

Variables, which are given at the input of `FB_init`, could be online changed after this method has been called. Normally this method can be used to dynamically make connections to multiple Q-SYS cores.

Example:

Switch the target server from “192.168.0.110” to “192.168.0.100”:

```
PROGRAM MAIN
VAR
    fbConnect           : FB_Connect('', '192.168.1.110', T#15S, fbResExtract);
    fbResExtract        : FB_QRC_ResExtract;
    nStep               : INT;
    bChangeTarget       : BOOL;
END_VAR
```

```
CASE nStep OF
    0:
        fbConnect.Connect();
        IF NOT fbConnect.bBusy AND NOT fbConnect.bError THEN
            nStep := nStep + 1;
        END_IF
    1:
        IF bChangeTarget THEN
            bChangeTarget := FALSE;
            nStep := nStep + 1;
        ELSE
            nStep := 3;
        END_IF
    2:
        fbConnect.FB_exit(FALSE);
        fbConnect.FB_init(FALSE, FALSE, '', '192.168.0.100', T#15S, fbResExtract);
        nStep := 0;
    3:
        (*Rest of Codes*)
END_CASE
```

4.1.2 FB_QRC_ResExtract

This function block enables the extraction of the received QRC frames.

This extraction function block is only designed for QRC response frames of the following QRC commands:

- Command Status.Get [▶ 24]
- Control-related commands (Control.Set [▶ 25] & Control.Get [▶ 25])
- Component-related commands (Component.Set [▶ 27] & Component.Get [▶ 27])
- "Change Control"-related commands (All methods of FB_QRC_ChangeGroup [▶ 28])
- Snapshot component (More information can be found at section Control button 'Load' of snapshot component [▶ 51] and Snapshot state and related properties [▶ 52])

The response frames of other QRC commands can be directly fetched with the property `aRxFrame`.

Syntax

```
FUNCTION_BLOCK FB_QRC_ResExtract IMPLEMENTS I_ResExtract
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Is set if an error occurs during the execution of the function block. Error details are located in the "Error List" window.
ipResult Message	I_TcMessage	Enables error handling with the `Tc3_EventLogger`.

Methods

Name	Description
ResExtract	Extract received QRC response frames.
Clear	Clear the internal memory.

Properties

Properties	Type	Access	Description
aCtrlProp	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_ControlEx	Get	Extracted control properties can be get with this property.
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Set, Get	Extracting QRC frames can be set or get with this property.
sEngineStatus	T_MaxString	Get	Q-SYS device information can be get with this property.

Interface

Name	Description
I_ResExtract	The interface that defines the extraction method.

Also see about this

- The attribute bSavOldRes [▶ 19]

4.1.2.1 sEngineStatus

This property enables to query the status information of the Q-SYS device.

Syntax

```
PROPERTY sEngineStatus : T_MaxString
```

4.1.2.2 Clear

This method enables clearing all saved snapshot properties that were queried via Poll [▶ 30] or AutoPoll [▶ 31].

Syntax

```
METHOD Clear : BOOL
```

This method is meaningful, if the used snapshot is obsolete. Read the section Workflow about extraction of snapshot properties [▶ 20] for more information.

4.1.2.3 I_ResExtract

METHODS

ResExtract: Extract received QRC response frames from Q-SYS device.

PROPERTIES

Properties	Type	Access	Description
aCtrlProp	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_ControlEx	Get	Get the extracted control properties with this property.
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Set, Get	QRC frames to be extracted can be set or get with this property.

aCtrlProp

List of control properties that has been extracted by ResExtract.

```
PROPERTY aCtrlProp       : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_ControlEx
```

arrRxFrame

QRC response frame that is ready to be extracted can be set or get with this property.

As mentioned [▶ 16] before, this function block can extract limited types of QRC response frames. The response frame that cannot be extracted by function block can be fetched with "getter" function before extraction. Furthermore, users can also write down their own QRC frame at “setter” function, in order to extract information from their own QRC frame.

```
PROPERTY aRxFrame        : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString
```

4.1.2.3.1 ResExtract

ResExtract

This method enables to extract control properties from a QRC response frame.

Syntax

```
METHOD ResExtract : BOOL
VAR_INPUT
    bSavOldRes      : BOOL;
END_VAR
```

VAR_INPUT

bSavOldRes: This variable determines whether the referenced function block will save past control properties that has been extracted from previous QRC frames. More information can be found at section “The attribute bSavOldRes [▶ 19]”.

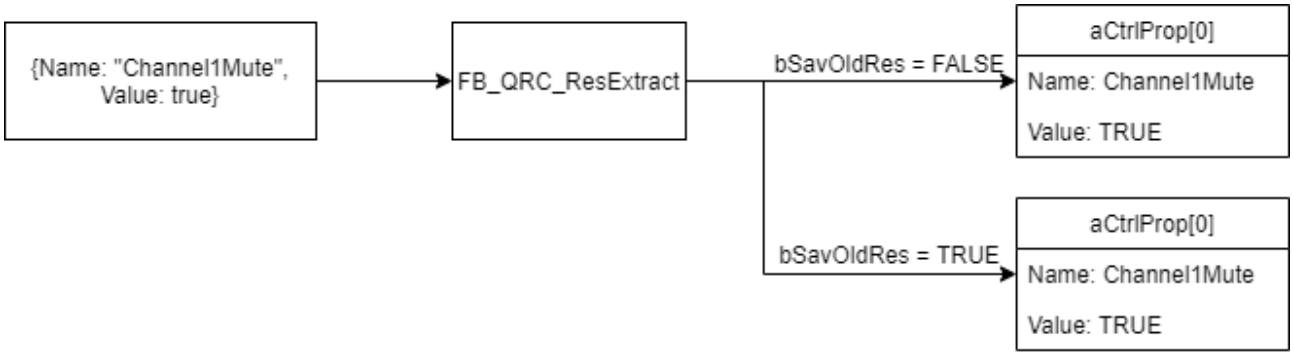
The attribute bSavOldRes

The input variable `bSavOldRes` of method `ResExtract` has been implemented to enable a configuration of received controls' information. The array `aCtrlProp` is able to store QRC_NUMBER_OF_CONTROL number of controls' information. This attribute can be changed in parameter list [▶ 42].

- By setting the attribute `bSavOldRes` to TRUE, all past controls' information will be stored. If upcoming controls' information which is already stored, the old controls' information will be overwritten by the new's.
- By setting the attribute `bSavOldRes` to FALSE, all past controls' information which were stored in the array `aCtrlProp` will be cleared. Only the latest controls' information will be stored.

To get a better understanding of the behavior, there is an example shown underneath.

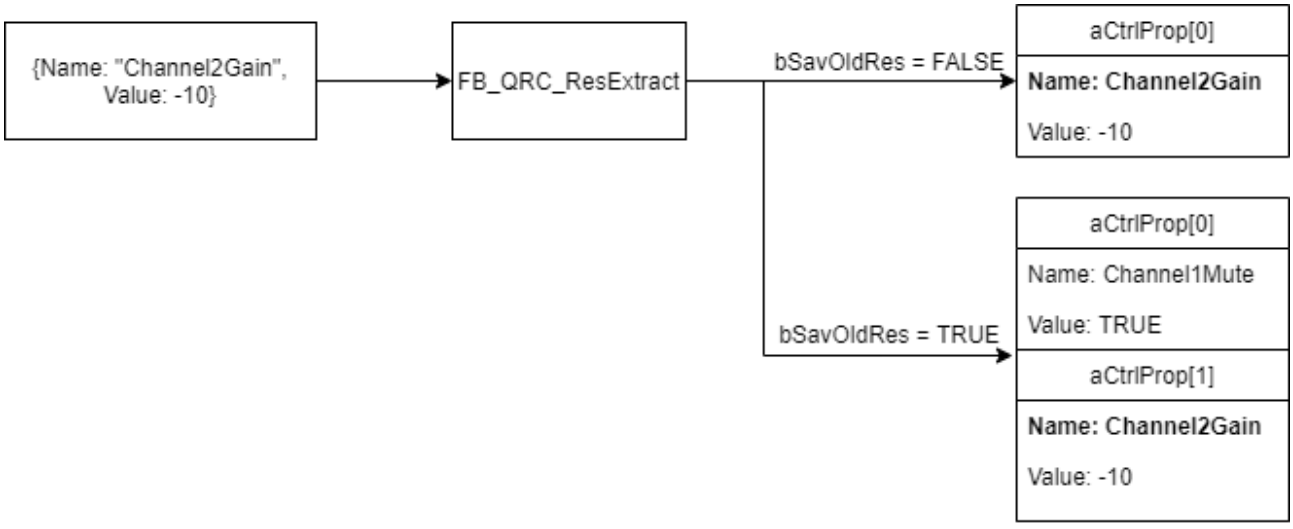
1st. Step: Control information of "**Channel1Mute**" received.



At Step 1, a QRC frame was received at `aRxFrame` and the control information are extracted by FB_QRC_RecExtract [▶ 16].

The array `aCtrlProp` is empty. Because of this, control **Channel1Mute** is saved at element `aCtrlProp[0]` whether `bSavOldRes` is TRUE or not.

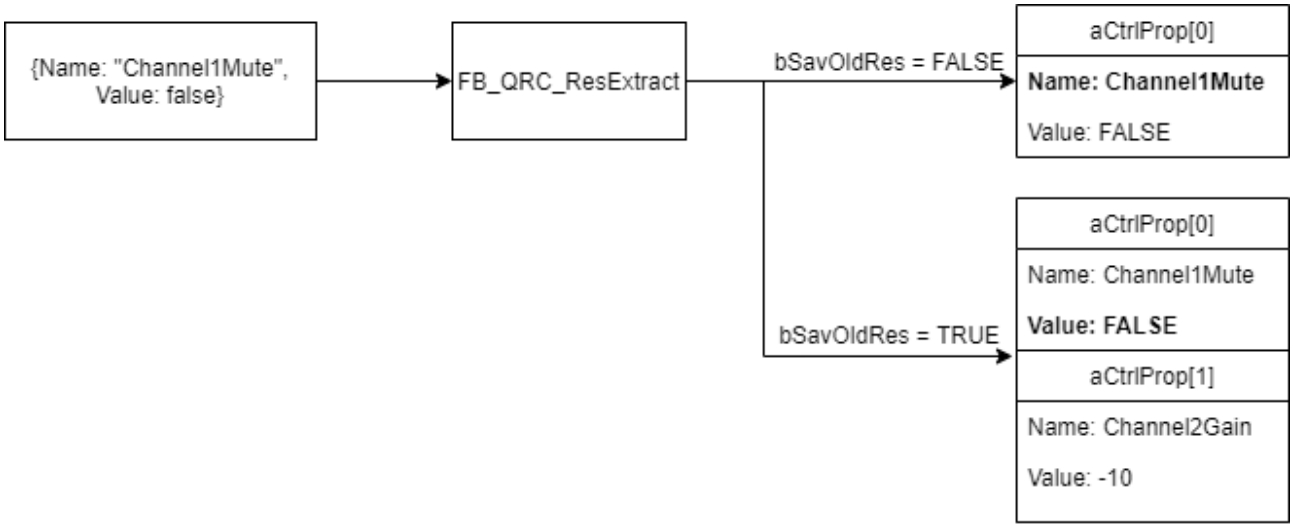
2nd Step: Control information of "**Channel2Gain**" (different control) received.



At Step 2, second QRC frame was received. After extraction of the new control information, it is stored depending on the value of `bSavOldRes`.

- If `bSavOldRes` is TRUE, the control **Channel2Gain** is stored at element `aRecProp[1]` because `aCtrlProp[0]` has stored another control Channel1Mute.
- If `bSavOldRes` is FALSE, the control **Channel2Gain** is stored at element `aCtrlProp[0]`. The control information of Channel1Mute which was stored at the same element will be overwritten.

3rd Step: Control information of "**Channel1Mute**" (An update of already received control) received.



At step 3, third QRC frame was received. After extraction, it recognized that the control name has been already stored at `aCtrlProp[0]`:

- If `bSavOldRes` is TRUE, the new-coming information of **Channel1Mute** will be stored at element `aRecProp[0]`. As a result of this, the stored control information of **Channel1Mute** gets updated and control information which stored at `aCtrlProp[1]` is kept.
- If `bSavOldRes` is FALSE, the new-coming control information of **Channel1Mute** will be stored at element `aCtrlProp[0]`. Other stored information will be cleared.

All past controls' properties will be saved only when the `bSavOldRes` is TRUE. In the case `bSavOldRes` is FALSE, all past control information will be cleared.

Workflow about extraction of snapshot properties

There are two ways to query a snapshot state, manually querying with method GetSnapshotState [▶ 40], or joining a change group and polling its changes. Based on the working principle of a change group, the polling function will only report to the changed control within a polling cycle. In some cases, it is impossible to determine a snapshot state. (e.g. a snapshot changes from "loaded" to "changed", then the Q-SYS device will only report that the control "`match`" changed from "true" to "false". The other related control "`last`" remains "true".) However, each time the method GetSnapshotState [▶ 40] is used, every related control of a requested snapshot will be queried. With the complete information the snapshot state can always be determined.

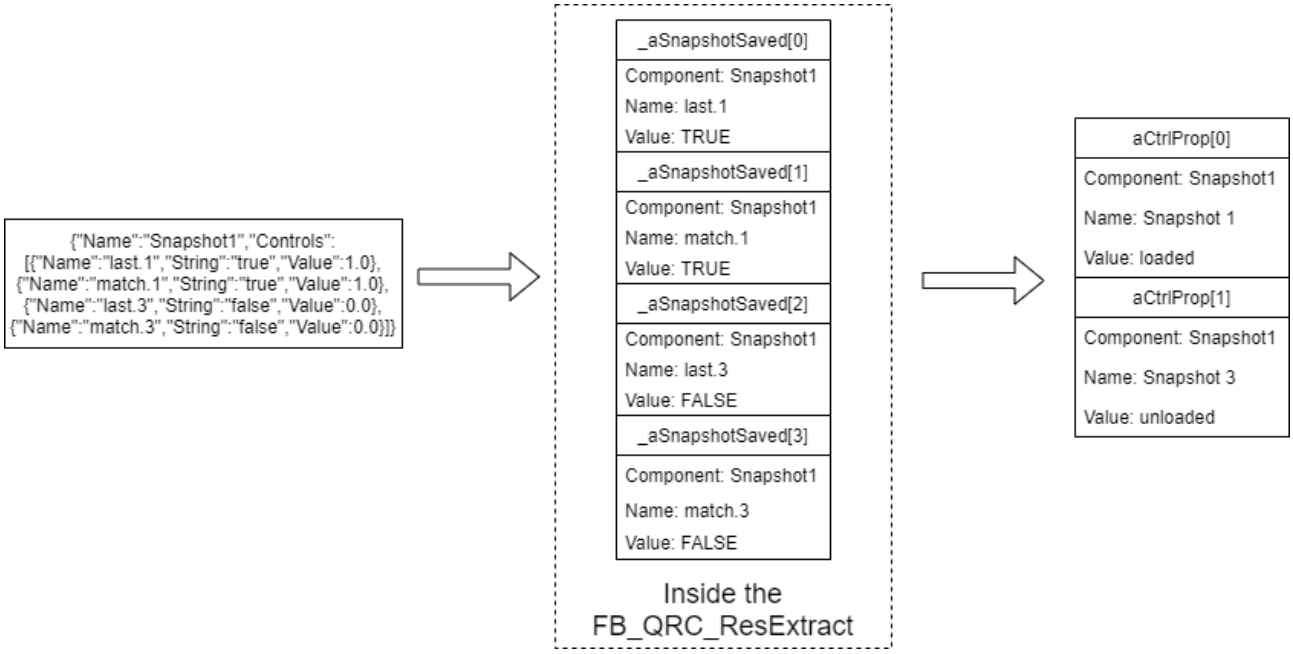
Because of the fact that each snapshot property which is queried by a polling function(Poll [▶ 30] or AutoPoll [▶ 31]), is stored internally, the Clear [▶ 30] method of the function block FB_QRC_ResExtract [▶ 16] can be used to release this storage.

After a response frame by a Q-SYS device arrived, all of snapshot controls' properties, which are queried by polling method, will be stored internally. (The attribute `bSavOldRes` has NO impact on this.) The snapshot control properties will be updated. With the help of the Clear method these properties can be deleted.

This logic has no impact to the `bSavOldRes` logic, which was described in the section Attribute `bSavOldRes`. However users can also set `bSavOldRes` to **TRUE** to save control properties at `aCtrlProp`.

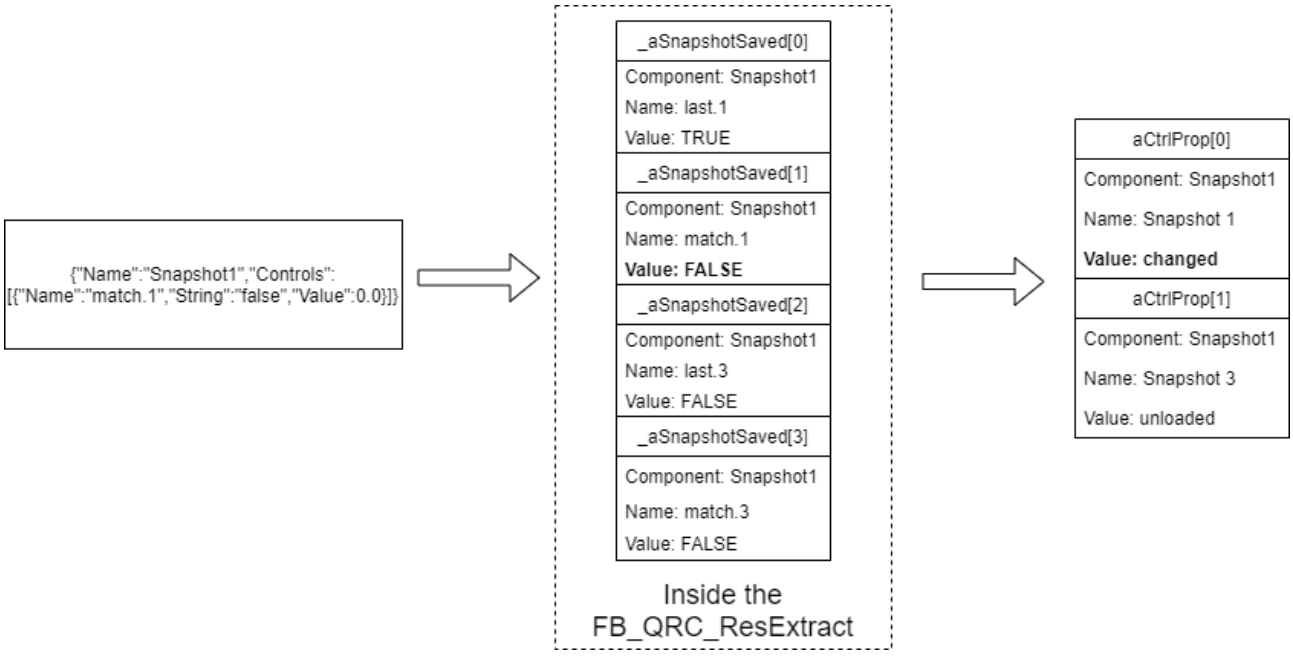
To get a better understanding of this behavior, there is an example shown underneath.

Step 1: After snapshot 1 and 3 (Name of Snapshot Bank is "Bank1", name of snapshot component is "Snapshot1".) has joined in the change group ("ChangeGroup 1"), the response frame was received:



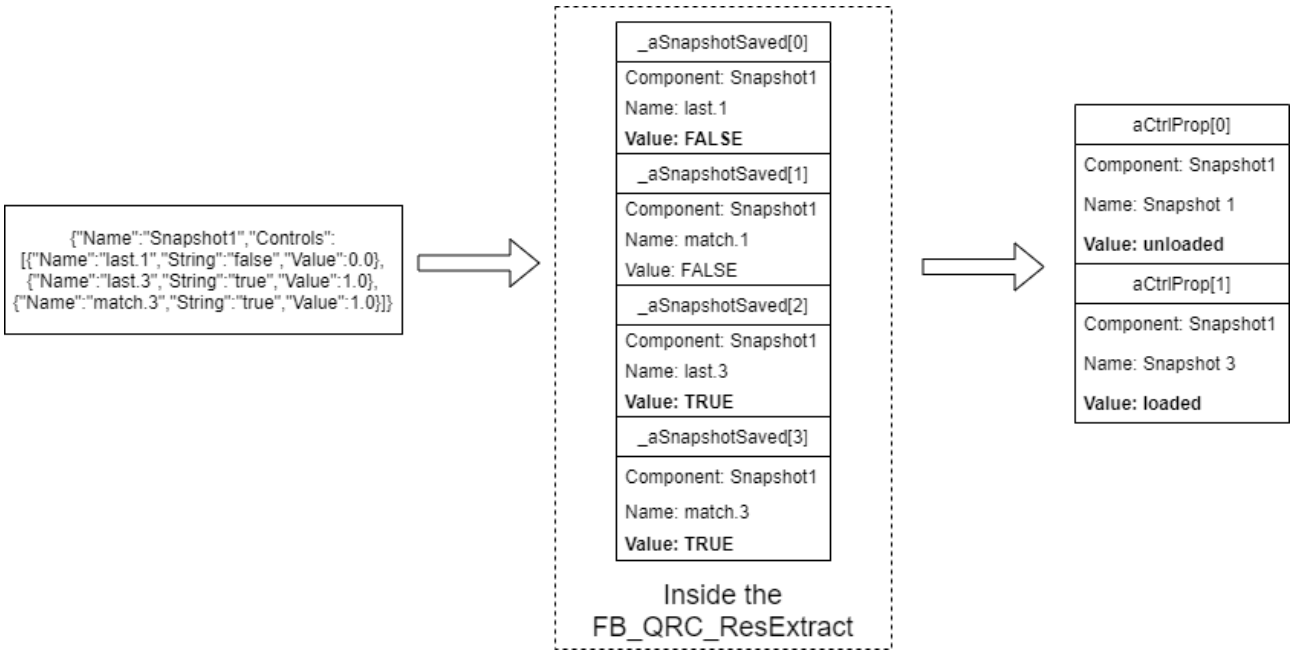
All related informations will be stored internally in an array. Snapshots' states are determined.

Step 2: In case some snapshot contained controls were changed within a polling cycle, a polling frame is arrived:



The property "`match.1`" will be updated in the internal array and the snapshot "Snapshot 1" changes its state from "loaded" to "changed". (`aCtrlProp[0]`)

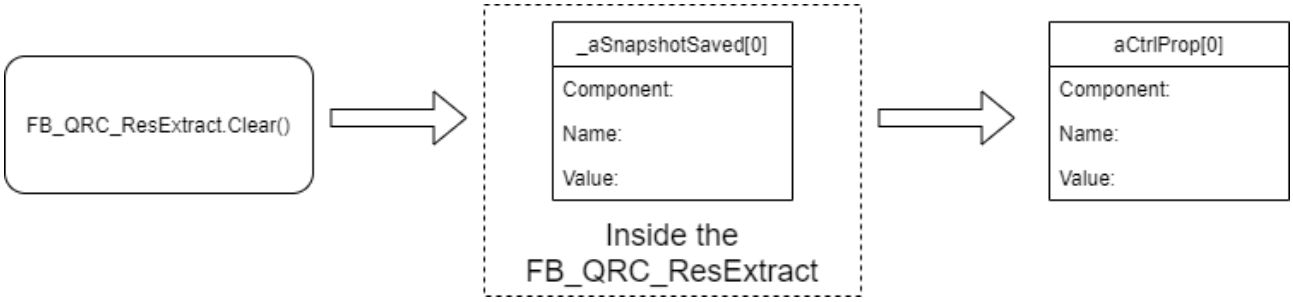
Step 3: Snapshot 3 is triggered.



The "Snapshot 3" was just triggered and the polling frame was received. Related control properties will be updated.

Step 4: Clear the internal array.

If users want to poll another snapshots' state and the stored properties are no longer useful, the method Clear [▶ 30] should be used to reset the internal array.



After the clear operation the internal array and the array `aCtrlProp` are both empty.

4.1.3 QRC Commands

In following paragraphs, 6 function blocks, which are located in the folder "QRC_Application" of library `Tc3_Qrc`, are designed based on the QRC specification. Each function block has a same method FB_init [▶ 23], and each method that implemented QRC specification has the same return type I_Connect [▶ 15].

4.1.3.1 FB_QRC_BasicCommand

This function block enables the coding of a QRC basic command.

Syntax

```
FUNCTION_BLOCK FB_QRC_BasicCommand
VAR_OUTPUT
    bError           : BOOL;
    ipResultMessage  : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
Logon	Log on Q-SYS device.
NoOp	Maintain TCP connection.
StatusGet	Get current status of the Q-SYS device.

4.1.3.1.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.1.2 LogOn

This method enables to log on the Q-SYS device.

Syntax

```
METHOD LogOn : I_Connect
VAR_INPUT
    sUserName               : STRING;
    nPassword               : UDINT;
END_VAR
```

VAR_INPUT

sUserName: User name.

nPassword: Password.

4.1.3.1.3 NoOp

This method enables to keep a TCP connection alive.

Syntax

```
METHOD NoOp : I_Connect
```

In FB_Connect, this method is internally used to keep TCP connection alive. The keep-alive cycle time is 45 second.

4.1.3.1.4 StatusGet

This method enables to query status information of the Q-SYS device.

Syntax

```
METHOD StatusGet : I_Connect
```

This method is automatically deployed by the Q-SYS device to return its status information whenever a client has been connected to the Q-SYS device or the state of the Q-SYS device changed. This status information can be easily extracted and fetched by the function block FB_QRC_ResExtract.

4.1.3.2 FB_QRC_Control

This function block enables the coding of QRC frames that are used to set or get control properties via Named Controls.

Syntax

```
FUNCTION_BLOCK FB_QRC_Control
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
Get	Get control properties via Named Control.
Set	Set properties of a control via Named Control.

4.1.3.2.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.2.2 Get

This method enables encoding of the QRC frames which are used for getting controls' properties via Named Controls.

Syntax

```
METHOD Get : I_Connect
VAR_INPUT
aControlName           : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF STRING;
END_VAR
```

VAR_INPUT

aControlName: List of target Named Controls that will be queried.

Named Controls should be listed starting from the first element of the array `aControlName`.

4.1.3.2.3 Set

This method enables encoding of the QRC frames which are used for setting control properties via Named Control.

Syntax

```
METHOD Set : I_Connect
VAR_INPUT
    stControlValue          : ST_Control;
END_VAR
```

VAR_INPUT

stControlValue: Properties of the target Named Control

4.1.3.3 FB_QRC_Component

This function block enables the coding of the QRC frames that are used to set/get control properties via Named Component. It is also used to list all existing components via Named Component.

Definition of Named Component: Named Component is a component control with a unique name property.

Syntax

```
FUNCTION_BLOCK FB_QRC_Component
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
Set	Set control properties via a Named Component.
Get	Get control properties via a Named Component.
GetComponent	Get control properties of all existing Named Components in a Q-SYS design.

4.1.3.3.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.3.2 Set

This method enables encoding a QRC frame that is used for setting one or more controls' properties of a Named Component.

Syntax

```
METHOD Set : I_Connect
VAR_INPUT
    sComponentName              : STRING;
    aControlValue               : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_Control;
END_VAR
```

VAR_INPUT

sComponentName: Name property of target Named Component.

aControlValue: Target controls' properties of the Named Component.

> Controls' properties should be listed starting from the first element of the array `aControlValue`.

4.1.3.3.3 Get

This method enables encoding a QRC frame that is used for getting one or more controls' properties on a Named Component.

Syntax

```
METHOD Get : I_Connect
VAR_INPUT
    sComponentName              : STRING;
    aControlName                : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF STRING;
END_VAR
```

VAR_INPUT

sComponentName: Name property of target Named Component.

aControlName: Target controls' name of the Named Component.

> Controls' name should be listed starting from the first element of the array `aControlName`.

4.1.3.3.4 GetComponent

This method enables encoding a QRC frame that is used to get controls' properties of all available Named Components.

Syntax

```
METHOD GetComponent : I_Connect
```

> 1. The response frame of this command cannot be extracted by FB_QRC_ResExtract [▶ 16].
> 2. Normally, the response frame of this command is extremely long (because each control of each Named component will be presented), please be aware of the buffer size QRC_BUFFER_SIZE [▶ 42].

4.1.3.4 FB_QRC_ChangeGroup

This function block enables the coding of the QRC frames that are used to edit or poll a change group.

Syntax

```
FUNCTION_BLOCK FB_QRC_ChangeGroup
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
AddControl	Add one or more controls to a change group via Named Controls.
AddComponent Control	Add one or more controls to a change group via Named Component.
Remove	Remove one or more controls from a change group.
Poll	Poll a change group to get its changes.
Destroy	Delete a change group.
Clear	Delete all controls from a change group.
Invalidate	Specify to all controllers to report their properties in the next polling round.
AutoPoll	Set up automatic polling.
AddSnapshot Control	Add one or more snapshots to a change group via Named Snapshot Component.

1. A change group is a grouping of Named Controls or Named Components. This function block is used to get more control properties with only one QRC frame.
2. If there is no target change group, it will be created automatically after the first `AddControl`, `AddComponentControl` or `AddSnapshotControl` command is received from the Q-SYS device.

4.1.3.4.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.4.2 AddControl

This method enables encoding a QRC frame that is used to add one or more controls via Named Control in a change group.

Syntax

```
METHOD AddControl : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
    aControlName              : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

aControlName: List of target Named Controls.

Controls' names should be listed starting from the first element of the array `aControlName`.

4.1.3.4.3 AddComponentControl

This method enables encoding a QRC frame that is used to add one or more controls within a Named Component in a change group.

Syntax

```
METHOD AddComponentControl : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
    sComponentName            : STRING;
    aControlName              : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

sComponentName: Name property of target Named Component.

aControlName: Target controls' names of the Named Component.

Controls' name should be listed starting from the first element of the array `aControlName`.

4.1.3.4.4 Remove

This method enables encoding a QRC frame that is used to remove one or more Named Controls from a change group.

Syntax

```
METHOD Remove : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
    aControlName              : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

aControlName: Target controls' names.

Controls' name should be listed starting from the first element of the array `aControlName`.

4.1.3.4.5 Poll

This method enables encoding a QRC frame that is used to poll a change group.

Syntax

```
METHOD Poll : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

4.1.3.4.6 Destroy

This method enables encoding a QRC frame that is used to destroy a change group. This change group will no longer exist.

Syntax

```
METHOD Destroy : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

Difference between `Destroy` method and `Clear` method:

`Clear` is used to delete all Named Controls / Name Components from a change group. This change group is still existed but empty.

`Destroy` is used to delete a change group. This change group will no longer exist after this operation.

4.1.3.4.7 Clear

This method enables encoding a QRC frame that is used to delete all Named Controls / Name Components from a change group. This change group is still existing.

Syntax

```
METHOD Clear : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

> Difference between `Destroy` method and `Clear` method:
>
> `Clear` is used to delete all Named Controls / Name Components from a change group. This change group is still existed but empty.
>
> `Destroy` is used to delete a change group. This change group will no longer exist after this operation.

4.1.3.4.8 Invalidate

This method enables encoding a QRC frame that is used to set all Named Controls / Name Components to "Dirty" state.

Syntax

```
METHOD Invalidate : I_Connect
VAR_INPUT
    sChangeGroupId            : STRING;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

> How the change group works internally in Q-SYS
>
> After a new control has been added to a change group, this control is marked as a “Dirty” state, which means that its current properties are not reported. Once its current properties are reported by Poll [▶ 30] or AutoPoll [▶ 31] methods, its state will change to “Clean”. Only the control which has the "Dirty" state will be reported by polling method, and the control that has the “Clean” state will not be reported. The control state will be switched from “Clean” to “Dirty” only if the properties of this control are changed.
>
> This method enables to set each control within a change group into “Dirty” state. It forces all controls to report their current state information by next Poll [▶ 30] or AutoPoll [▶ 31] method.

4.1.3.4.9 AutoPoll

This method enables encoding a QRC frame that is used to set all Named Controls / Name Components to the "Dirty" state.

Syntax

```
METHOD Poll : I_Connect
VAR_INPUT
    sChangeGroupId        : STRING;
    fRate                 : REAL;
END_VAR
```

VAR_INPUT

sChangeGroupId: Change group ID.

fRate: Polling interval in seconds. The minimum value of it is 0.1s.

4.1.3.4.10 AddSnapshotControl

This method enables the encoding of a QRC frame that is used for joining multiple snapshots in a change control.

> The snapshot control can not be joined in a change group via Named Control. In this version the snapshot related sub-controls are joined in a change group via **Named Component**.

Syntax

```
METHOD AddSnapshotControl : I_ResExtract
VAR_INPUT
    sChangeGroupId           : STRING;
    sComponentName           : STRING;
    aSnapshotNr              : ARRAY [0..23] OF USINT;
END_VAR
```

sChangeGroupId: Change Group Id.

sComponentName: Name of the snapshot component.

aSnapshotNr: Array of target snapshot sequence number.

4.1.3.5 FB_QRC_Mixer

This function block allows several different values to be set on a named mixer.

Definition of a named mixer: a mixer component with a unique name.

Syntax

```
FUNCTION_BLOCK FB_QRC_ChangeGroup
VAR_OUTPUT
    bError                       : BOOL;
    ipResultMessage              : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
SetCrossPointGain	Set crosspoint gain value for mixer inputs and outputs.
SetCrossPointDelay	Set crosspoint delay for mixer inputs and outputs.
SetCrossPointMute	Mute or unmute crosspoint for mixer inputs and outputs.
SetCrossPointSolo	Enable or disable crosspoint solo for mixer inputs and outputs.
SetInputGain	Set gain for mixer inputs.
SetInputMute	Mute or unmute mixer inputs.
SetInputSolo	Enable or disable solo for mixer inputs.
SetOutputGain	Set gain for mixer outputs.
SetOutputMute	Mute or unmute mixer outputs.
SetCueMute	Mute or unmute mixer cues.
SetCueGain	Set gain for mixer cues.
SetInputCueEnable	Enable or disable cues for mixer inputs.
SetInputCueAfi	Enable or disable Cue-AFL (After Fader Level) for mixer inputs.

Example

The syntax supports either numbers separated by spaces or commas, ranges of numbers, or all numbers (*). It supports negation of the selection with the "!" operator.

Here are a few examples:

Input/output	Description
*	All
1 2 3	Channels 1, 2, 3
1-6	Channels 1 to 6
1-6 9	Channel 1 to 6 and 9
1-3 5-9	Channel 1 to 3 and 5 to 9
1-8 !3	Channel 1 to 8 except 3
* !3-5	All except channels 3 to 5

4.1.3.5.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.5.2 SetCrossPointGain

This method enables encoding a QRC frame that is used to set the crosspoint gain value for the inputs and outputs of a named mixer.

Syntax

```
METHOD SetCrossPointGain : I_Connect
VAR_INPUT
    stCrossSpec          : ST_CrossSpec;
END_VAR
```

VAR_INPUT

stCrossSpec: Crosspoint gain properties.

4.1.3.5.3 SetCrossPointDelay

This method enables encoding a QRC frame that is used to set the crosspoint delay value for inputs and outputs of a named mixer.

Syntax

```
METHOD SetCrossPointDelay : I_Connect
VAR_INPUT
    stCrossSpec           : ST_CrossSpec;
END_VAR
```

VAR_INPUT

stCrossSpec: Crosspoint delay properties.

4.1.3.5.4 SetCrossPointMute

This method enables encoding a QRC frame that is used to set the crosspoint muted or unmuted for inputs and outputs of a named mixer.

Syntax

```
METHOD SetCrossPointMute : I_Connect
VAR_INPUT
    stCrossSpec          : ST_CrossSpec;
END_VAR
```

VAR_INPUT

stCrossSpec: Crosspoint mute properties.

4.1.3.5.5 SetCrossPointSolo

This method enables encoding a QRC frame that is used to enable or disable crosspoint solo for inputs and outputs of a named mixer.

Syntax

```
METHOD SetCrossPointSolo : I_Connect
VAR_INPUT
    stCrossSpec          : ST_CrossSpec;
END_VAR
```

VAR_INPUT

stCrossSpec: Crosspoint solo properties.

4.1.3.5.6 SetInputGain

This method enables encoding a QRC frame that is used to set gain value for inputs of a named mixer.

Syntax

```
METHOD SetInputGain : I_Connect
VAR_INPUT
    stInputSpec     : ST_InputSpec;
END_VAR
```

VAR_INPUT

stInputSpec: Input gain properties.

4.1.3.5.7 SetInputMute

This method enables encoding a QRC frame that is used to set inputs muted or unmuted of a named mixer.

Syntax

```
METHOD SetInputMute : I_Connect
VAR_INPUT
    stInputSpec     : ST_InputSpec;
END_VAR
```

VAR_INPUT

stInputSpec: Input mute properties.

4.1.3.5.8 SetInputSolo

This method enables encoding a QRC frame that is used to enable or disable solo for inputs of a named mixer.

Syntax

```
METHOD SetInputSolo : I_Connect
VAR_INPUT
    stInputSpec     : ST_InputSpec;
END_VAR
```

VAR_INPUT

stInputSpec: Input solo properties.

4.1.3.5.9 SetOutputGain

This method enables encoding a QRC frame that is used to set gain value for outputs of a named mixer.

Syntax

```
METHOD SetOutputGain : I_Connect
VAR_INPUT
    stOutputSpec     : ST_OutputSpec;
END_VAR
```

VAR_INPUT

stOutputSpec: Output gain properties.

4.1.3.5.10 SetOutputMute

This method enables encoding a QRC frame that is used to mute or unmute for outputs of a named mixer.

Syntax

```
METHOD SetOutputMute : I_Connect
VAR_INPUT
    stOutputSpec     : ST_OutputSpec;
END_VAR
```

VAR_INPUT

stOutputSpec: Output mute properties.

4.1.3.5.11 SetCueMute

This method enables encoding a QRC frame that is used to mute or unmute for mixer cues.

Syntax

```
METHOD SetCueMute : I_Connect
VAR_INPUT
    stCueSpec     : ST_CueSpec;
END_VAR
```

VAR_INPUT

stCueSpec: Cue mute properties.

4.1.3.5.12 SetCueGain

This method enables encoding a QRC frame that is used to set gain value for mixer cues.

Syntax

```
METHOD SetCueGain : I_Connect
VAR_INPUT
    stCueSpec     : ST_CueSpec;
END_VAR
```

VAR_INPUT

stCueSpec: Cue gain properties.

4.1.3.5.13 SetInputCueEnable

This method enables encoding a QRC frame that is used to enable or disable cues and inputs of named mixer.

Syntax

```
METHOD SetInputCueGain : I_Connect
VAR_INPUT
    stInputCueSpec     : ST_InputCueSpec;
END_VAR
```

VAR_INPUT

stInputCueSpec: Input and cue properties.

4.1.3.5.14 SetInputCueAfi

This method enables encoding a QRC frame that is used to enable or disable cue AFL (After Fader Level) for mixer inputs.

Syntax

```
METHOD SetInputCueAfi : I_Connect
VAR_INPUT
    stInputCueSpec    : ST_InputCueSpec;
END_VAR
```

VAR_INPUT

stInputCueSpec:Cue AFL properties.

4.1.3.6 FB_QRC_LoopPlayer

This function block enables the query of a file playback on a named loop player.

> Definition of a named loop player: a named loop player is a loop player component with a unique name.

Syntax

```
FUNCTION_BLOCK FB_QRC_LoopPlayer
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
Start	Start playback.
Stop	Stop playback.
Cancel	Cancel playback.

4.1.3.6.1 FB_init

Syntax

```
Method FB_init : BOOL
VAR_INPUT
    iConnect                : I_Connect;
END_VAR
```

VAR_INPUT

iConnect: The function block that implemented the interface I_Connect [▶ 15].

Example:

Declaration of the function block `FB_QRC_Control`:

```
PROGRAM MAIN
VAR
    fbResExtract        : FB_QRC_ResExtract;
    fbConnect           : FB_Connect('', '192.168.1.101', T#15S, fbResExtract);
    fbQrcControl        : FB_QRC_Control(fbConnect);
END_VAR
```

4.1.3.6.2 Start

This method enables to start playing on a named loop player.

Syntax

```
METHOD Start : I_Connect
VAR_INPUT
    stJobSpec            : ST_JobSpec;
END_VAR
```

VAR_INPUT

stJobSpec: Properties of the job that will be played back on a named loop player.

4.1.3.6.3 Stop

This method enables to stop playback on a named loop player.

Syntax

```
METHOD Stop : I_Connect
VAR_INPUT
    sName           : STRING;
    aOutput         : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF USINT;
    bLog            : BOOL := FALSE;
END_VAR
```

VAR_INPUT

sName: The name of the loop player.

aOutput: Array of output channels.

bLog: Optional attribute for event message, FALSE in default.

4.1.3.6.4 Cancel

This method enables to cancel a job on a named loop player.

Syntax

```
METHOD Cancel : I_Connect
VAR_INPUT
    sName           : STRING;
    aOutput         : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF USINT;
    bLog            : BOOL := FALSE;
END_VAR
```

VAR_INPUT

sName: The name of the loop player.

aOutput: Array of output channels.

bLog: Optional attribute for event message, FALSE in default.

4.1.3.7 FB_QRC_Snapshot

This function block enables the encoding of QRC frames that are used to load/save snapshots. In addition, it can be used to get multiple snapshot states.

This function block is available from version 1.1.0.0.
This command is not listed in the QRC specification.

- Before using this function block or related methods, first read the section Snapshot state and related properties [▶ 19].

Syntax

```
FUNCTION_BLOCK FB_QRC_Snapshot
VAR_OUTPUT
    bError                      : BOOL;
    ipResultMessage             : I_TcMessage;
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Set when an error has occurred. Error details are located in the **Error List** window.
ipResultMessage	I_TcMessage	Enables error handling with the Tc3_EventLogger.

Methods

Name	Description
FB_init	Initialization method
Load	Trigger a snapshot.
Save	Save a snapshot. (Overwrite old snapshot.)
GetSnapshotState	Get multiple snapshot states in one snapshot bank.

4.1.3.7.1 Load

This method enables the encoding of a QRC frame that is used for triggering a snapshot.

Syntax

```
METHOD Load : I_Connect
VAR_INPUT
    sBankName       : STRING;
    nSnapshotNr     : USINT;
    fRamp           : REAL := 0; (*Optional*)
END_VAR
```

sBankName: Name of the Snapshot Bank.

nSnapshotNr: Sequence number of the target snapshot.

fRamp: Optional ramp time.

> The **Name of a Snapshot Bank** is totally different to the **name of the snapshot component**. How to find the name of a Snapshot Bank is described in the section Easy way to find control name, component name and snapshot bank name [▶ 50].

4.1.3.7.2 Save

This method enables the encoding of a QRC frame that is used for saving a snapshot.

Syntax

```
METHOD Save : I_Connect
VAR_INPUT
    sBankName       : STRING;
    nSnapshotNr     : USINT;
    fRamp           : REAL := 0; (*Optional*)
END_VAR
```

sBankName: Name of the Snapshot Bank.

nSnapshotNr: Sequence number of the target snapshot.

fRamp: Optional ramp time.

The **Name of a Snapshot Bank** is totally different to the **name of the snapshot component**. How to find the name of a Snapshot Bank is described in the section Easy way to find control name, component name and snapshot bank name [▶ 50].

4.1.3.7.3 GetSnapshotState

Each snapshot state can be determined by two related controls within the snapshot component. This method enables the encoding of a QRC frame that is used for querying multiple snapshot states.

Syntax

```
METHOD GetSnapshotState : I_Connect
VAR_INPUT
    sComponentName      : STRING;
    aSnapshotNr         : ARRAY[0..23] OF USINT;
END_VAR
```

sComponentName: Name of the snapshot component.

aSnapshotNr: Array of requested snapshot sequence number.

1. The **name of the snapshot component** is totally different to the **name of a Snapshot Bank**. How to find the name of the snapshot component is described in the section Easy way to find control name, component name and snapshot bank name [▶ 50].
2. Set the attribute `bSavOldRes` to TRUE of function block `FB_QRC_RecExtract` to save the past snapshot states at property `aCtrlProp`.

4.2 Structures, enumerations, GVL

4.2.1 E_FileMode

```
Type E_FileMode
{
    mono,
    stereo
} USINT;
END_TYPE
```

4.2.2 ST_Control

```
TYPE ST_CONTROL
STRUCT
    sName         : STRING := '';  (*Name of Named Control*)
    sValue        : STRING := '';  (*Value of Named Control*)
    sString       : STRING := '';  (*String of Named Control*)
    fRamp         : REAL   := 0;   (*Optional ramp time of Named Control*)
END_STRUCT
END_TYPE
```

4.2.3 ST_ControlEx

This structure extends St_Control [▶ 40] and it is designed only for the property `arrCtrlProp` of function block FB_QRC_ResExtract [▶ 16].

```
TYPE ST_ControlEx EXTENDS ST_Control
STRUCT
    sComponent       : STRING := '';  (*Component name*)
    fPosition        : STRING := '';  (*Control position*)
END_STRUCT
END_TYPE
```

4.2.4 Structure about Mixer

ST_CrossSpec

```
TYPE ST_CrossSpec:
STRUCT
    sName           : STRING := '';  (*Name of named mixer*)
    sInputs         : STRING := '';  (*Input channel of named mixer*)
    sOutputs        : STRING := '';  (*Output channel of named mixer*)
    sValue          : STRING := '';  (*value of named mixer*)
    fRamp           : REAL   := 0;   (*Optional ramp time of named mixer*)
END_STRUCT
END_TYPE
```

ST_InputSpec

```
TYPE ST_InputSpec:
STRUCT
    sName           : STRING := '';  (*Name of named mixer*)
    sInputs         : STRING := '';  (*Input channel of named mixer*)
    sValue          : STRING := '';  (*value of named mixer*)
    fRamp           : REAL   := 0;   (*Optional ramp time of named mixer*)
END_STRUCT
END_TYPE
```

ST_OutputSpec

```
TYPE ST_OutputSpec:
STRUCT
    sName           : STRING := '';  (*Name of named mixer*)
    sOutputs        : STRING := '';  (*Output channel of named mixer*)
    sValue          : STRING := '';  (*value of named mixer*)
    fRamp           : REAL   := 0;   (*Optional ramp time of named mixer*)
END_STRUCT
END_TYPE
```

ST_CueSpec

```
TYPE ST_CueSpec:
STRUCT
    sName           : STRING := '';  (*Name of named mixer*)
    sCues           : STRING := '';  (*Cue of named mixer*)
    sValue          : STRING := '';  (*value of named mixer*)
    fRamp           : REAL   := 0;   (*Optional ramp time of named mixer*)
END_STRUCT
END_TYPE
```

ST_InputCueSpec

```
TYPE ST_InputCueSpec EXTENDS ST_CueSpec:
STRUCT
    sInputs         : STRING  := '';(*Input channel of named mixer*)
END_STRUCT
END_TYPE
```

4.2.5 ST_FileSpec

```
TYPE ST_FileSpec:
STRUCT
    sFileName    : T_MaxString;
    eMode        : E_FileMode;
    nOutput      : USINT; (*Output Channel*)
END_STRUCT
END_TYPE
```

4.2.6 ST_JobSpec

```
TYPE ST_JobSpec:
STRUCT
    sName           : STRING :='';
    nStartTime         : UDINT  := 0;
    aFiles             : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_FileSpec;
```

```
    bLoop               : BOOL;
    nSeek               : UDINT := 0;
    bLog                : BOOL;
END_STRUCT
END_TYPE
```

4.2.7 Param

Name	Default value	Description
QRC_RECEIVE_POLLING_TIME	100ms	Polling time for the TCP connection
QRC_RECEIVE_TIMEOUT	10s	Time for receiver timeout
QRC_BUFFER_SIZE	2500	QRC frame buffer size in byte
QRC_NUMBER_OF_CONTROL	50	The maximum number of controls that are allowed to send, to receive and to extract.

1. QRC_BUFFER_SIZE defined the length of sending buffer `sTxFrame`. Before each QRC frame is transmitted, this QRC frame was measured and checked whether the length of this frame is greater than QRC_BUFFER_SIZE. In some cases (e.g. Hundreds of controls are transmitted via Control.Set [▶ 25] or Component.Set [▶ 27]) the buffer is easily overloaded. If the buffer has been overflowed, an error "Buffer overflowed" will occur before transmitting this QRC frame. This QRC frame will be ignored until the value of QRC_BUFFER_SIZE has been increased.

2. QRC_BUFFER_SIZE is considered to be highly relevant to QRC_NUMBER_OF_CONTROL. In consequence, the value of QRC_BUFFER_SIZE has to be changed accordingly. (The Proportion 1:50 (1 control - 50 Byte) is recommended).

4.3 Interfaces

4.3.1 I_Connect

METHODS

Connect: Create a TCP connection.

Disconnect: Terminate a TCP connection.

Send: Send QRC frames.

M_Receive: Receive QRC frames.

PROPERTIES

Properties	Type	Access	Description
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Get	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the received QRC response frame can be queried with this property.
sTxFrame	STRING(QRC_BUFFER_SIZE)	Set	As soon as the falling edge of `bBusy` occurs and `bError` is FALSE, the QRC frame to be sent can be set.

Connect

This method enables creating a TCP connection.

```
Method Connect            : BOOL
```

This process is finished as soon as the return value is TRUE.

Disonnect

This method enables terminating a TCP connection.

```
Method Disonnect          : BOOL
```

This process is finished as soon as the return value is TRUE.

Send

This method enables sending a QRC frame and to get the response frame from Q-SYS device automatically after sending.

```
Method Send               : I_ResExtract
```

This method is finished as soon as the falling edge of `bBusy` occurs and property `aRxFrame` is not empty. The response frame can be fetched at property `aRxFrame`.

Receive

This method enables receiving a QRC frame.

```
Method Receive            : I_ResExtract
```

This method is finished as soon as the falling trigger of `bBusy` is triggered and property. `aRxFrame` is not empty. The response frame can be fetched at property `aRxFrame`.

aRxFrame

List of received QRC response frames.

```
PROPERTY aRxFrame       : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString
```

sTxFrame

A QRC frame which is ready to send to Q-SYS device.

```
PROPERTY sTxFrame         : STRING(QRC_BUFFER_SIZE)
```

4.3.2 I_ResExtract

METHODS

ResExtract: Extract received QRC response frames from Q-SYS device.

PROPERTIES

Properties	Type	Access	Description
aCtrlProp	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_ControlEx	Get	Get the extracted control properties with this property.
aRxFrame	ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString	Set, Get	QRC frames to be extracted can be set or get with this property.

aCtrlProp

List of control properties that has been extracted by ResExtract.

```
PROPERTY aCtrlProp       : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF ST_ControlEx
```

arrRxFrame

QRC response frame that is ready to be extracted can be set or get with this property.

As mentioned [▶ 16] before, this function block can extract limited types of QRC response frames. The response frame that cannot be extracted by function block can be fetched with "getter" function before extraction. Furthermore, users can also write down their own QRC frame at “setter” function, in order to extract information from their own QRC frame.

```
PROPERTY aRxFrame        : ARRAY[0..QRC_NUMBER_OF_CONTROL] OF T_MaxString
```

4.3.2.1 ResExtract

ResExtract

This method enables to extract control properties from a QRC response frame.

Syntax

```
METHOD ResExtract : BOOL
VAR_INPUT
    bSavOldRes      : BOOL;
END_VAR
```

VAR_INPUT

bSavOldRes: This variable determines whether the referenced function block will save past control properties that has been extracted from previous QRC frames. More information can be found at section “The attribute bSavOldRes [▶ 44]”.

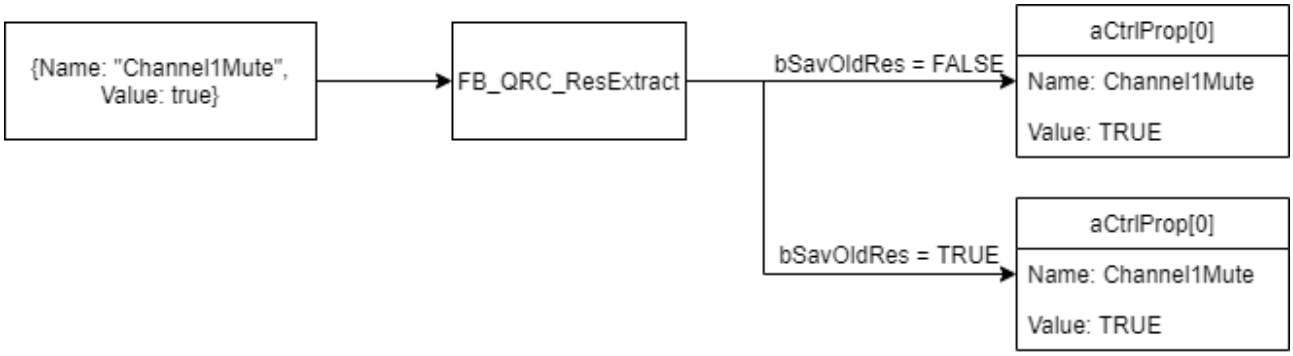
4.3.2.1.1 The attribute bSavOldRes

The input variable `bSavOldRes` of method `ResExtract` has been implemented to enable a configuration of received controls' information. The array `aCtrlProp` is able to store QRC_NUMBER_OF_CONTROL number of controls' information. This attribute can be changed in parameter list [▶ 42].

- By setting the attribute `bSavOldRes` to TRUE, all past controls' information will be stored. If upcoming controls' information which is already stored, the old controls' information will be overwritten by the new's.
- By setting the attribute `bSavOldRes` to FALSE, all past controls' information which were stored in the array `aCtrlProp` will be cleared. Only the latest controls' information will be stored.

To get a better understanding of the behavior, there is an example shown underneath.

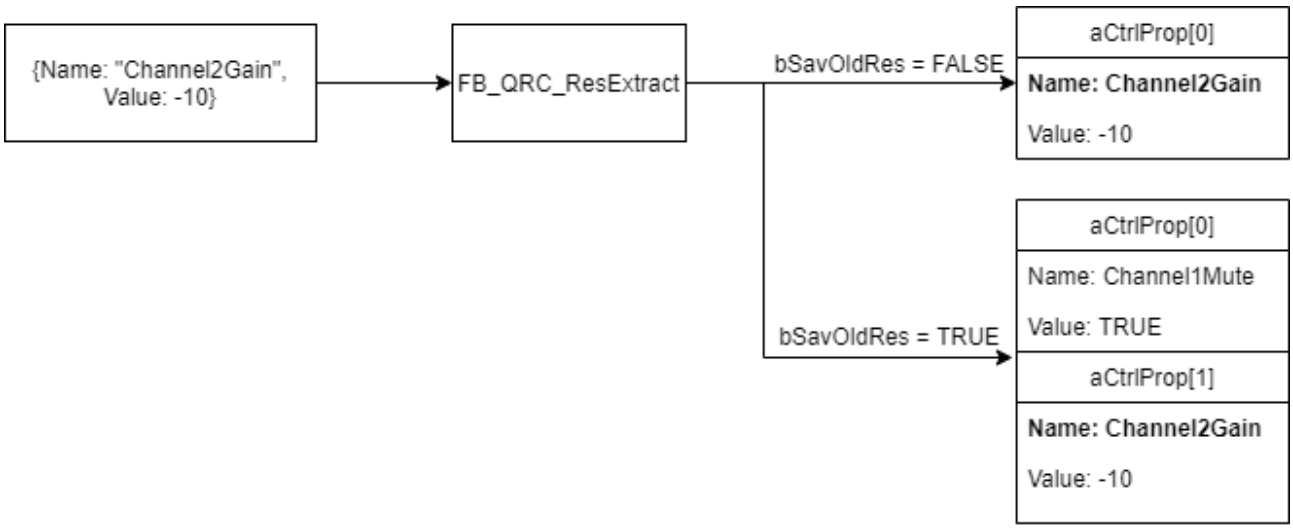
1st. Step: Control information of "**Channel1Mute**" received.



At Step 1, a QRC frame was received at `aRxFrame` and the control information are extracted by FB_QRC_RecExtract [▶ 16].

The array `aCtrlProp` is empty. Because of this, control **Channel1Mute** is saved at element `aCtrlProp[0]` whether `bSavOldRes` is TRUE or not.

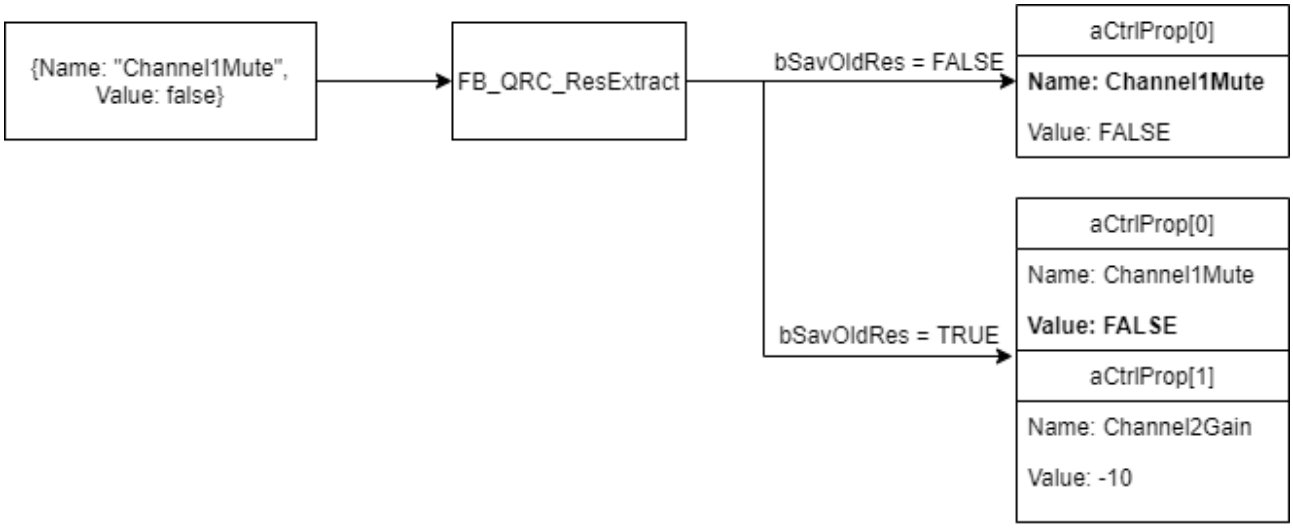
2nd Step: Control information of "**Channel2Gain**" (different control) received.



At Step 2, second QRC frame was received. After extraction of the new control information, it is stored depending on the value of `bSavOldRes`.

- If `bSavOldRes` is TRUE, the control **Channel2Gain** is stored at element `aRecProp[1]` because `aCtrlProp[0]` has stored another control Channel1Mute.
- If `bSavOldRes` is FALSE, the control **Channel2Gain** is stored at element `aCtrlProp[0]`. The control information of Channel1Mute which was stored at the same element will be overwritten.

3rd Step: Control information of "**Channel1Mute**" (An update of already received control) received.



At step 3, third QRC frame was received. After extraction, it recognized that the control name has been already stored at `aCtrlProp[0]`:

- If `bSavOldRes` is TRUE, the new-coming information of **Channel1Mute** will be stored at element `aRecProp[0]`. As a result of this, the stored control information of **Channel1Mute** gets updated and control information which stored at `aCtrlProp[1]` is kept.
- If `bSavOldRes` is FALSE, the new-coming control information of **Channel1Mute** will be stored at element `aCtrlProp[0]`. Other stored information will be cleared.

All past controls' properties will be saved only when the `bSavOldRes` is TRUE. In the case `bSavOldRes` is FALSE, all past control information will be cleared.

4.3.2.1.2 Workflow about extraction of snapshot properties

There are two ways to query a snapshot state, manually querying with method GetSnapshotState [▶ 40], or joining a change group and polling its changes. Based on the working principle of a change group, the polling function will only report to the changed control within a polling cycle. In some cases, it is impossible to determine a snapshot state. (e.g. a snapshot changes from "loaded" to "changed", then the Q-SYS device will only report that the control "`match`" changed from "true" to "false". The other related control "`last`" remains "true".) However, each time the method GetSnapshotState [▶ 40] is used, every related control of a requested snapshot will be queried. With the complete information the snapshot state can always be determined.

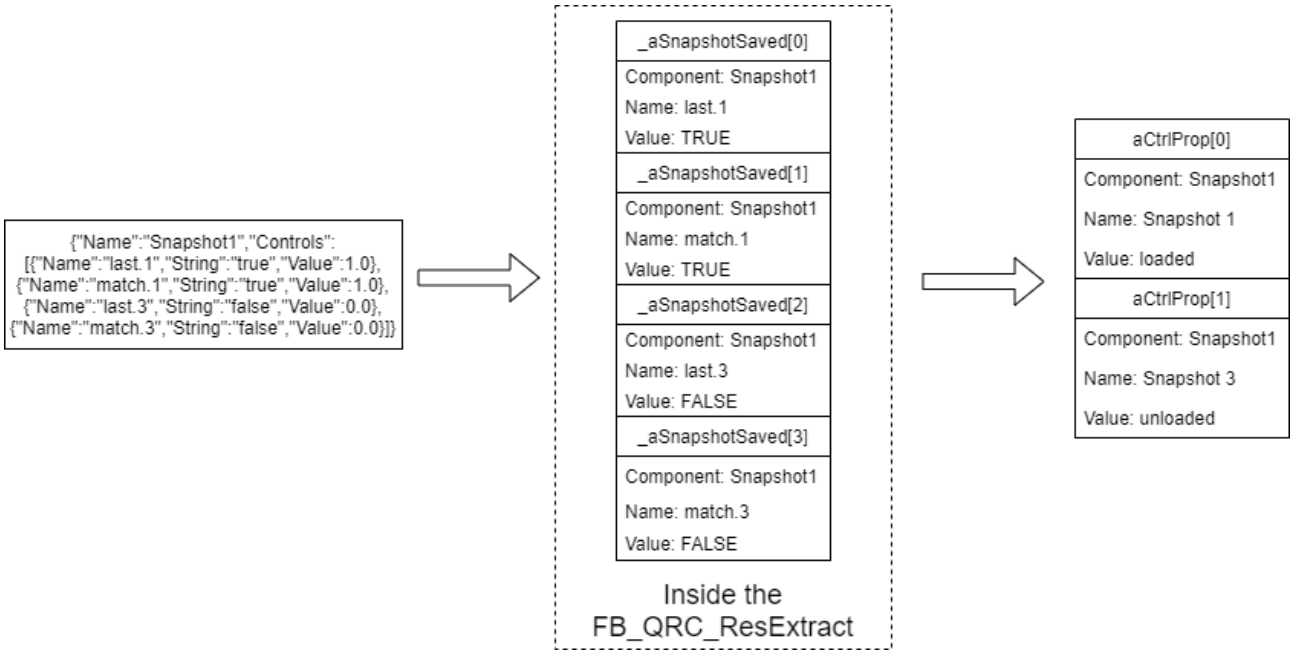
Because of the fact that each snapshot property which is queried by a polling function(Poll [▶ 30] or AutoPoll [▶ 31]), is stored internally, the Clear [▶ 30] method of the function block FB_QRC_ResExtract [▶ 16] can be used to release this storage.

After a response frame by a Q-SYS device arrived, all of snapshot controls' properties, which are queried by polling method, will be stored internally. (The attribute `bSavOldRes` has NO impact on this.) The snapshot control properties will be updated. With the help of the Clear method these properties can be deleted.

> This logic has no impact to the `bSavOldRes` logic, which was described in the section Attribute `bSavOldRes`. However users can also set `bSavOldRes` to **TRUE** to save control properties at `aCtrlProp`.

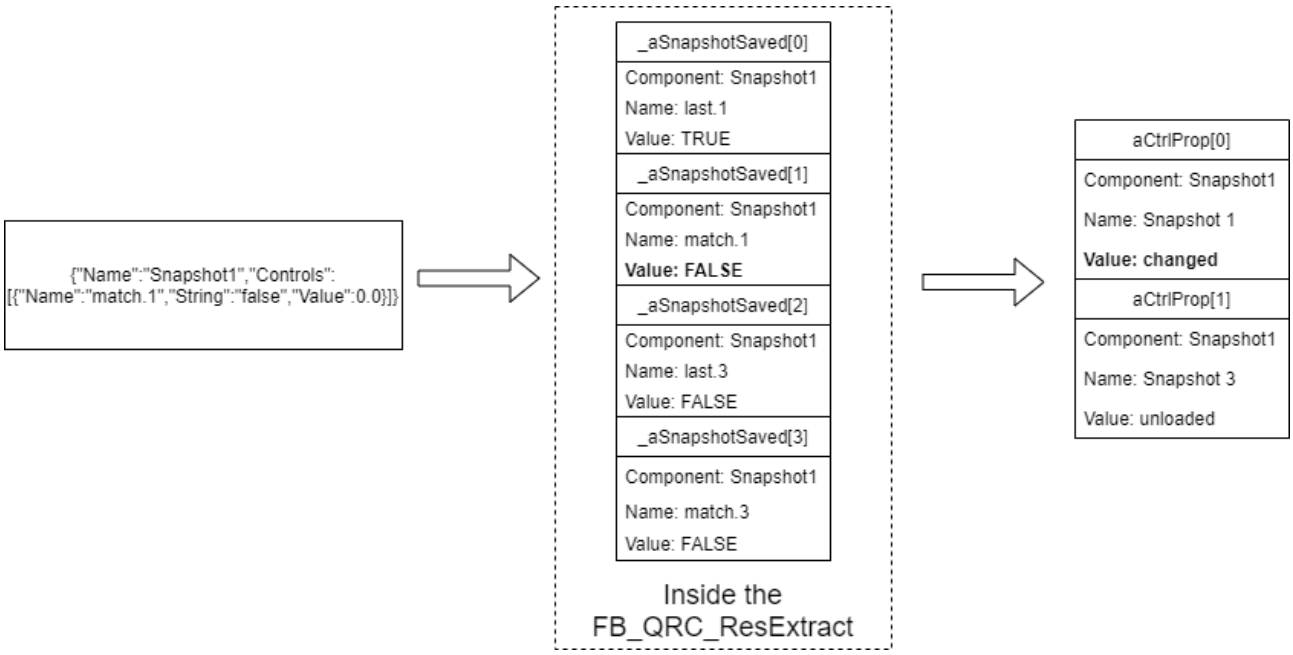
To get a better understanding of this behavior, there is an example shown underneath.

Step 1: After snapshot 1 and 3 (Name of Snapshot Bank is "Bank1", name of snapshot component is "Snapshot1".) has joined in the change group ("ChangeGroup 1"), the response frame was received:



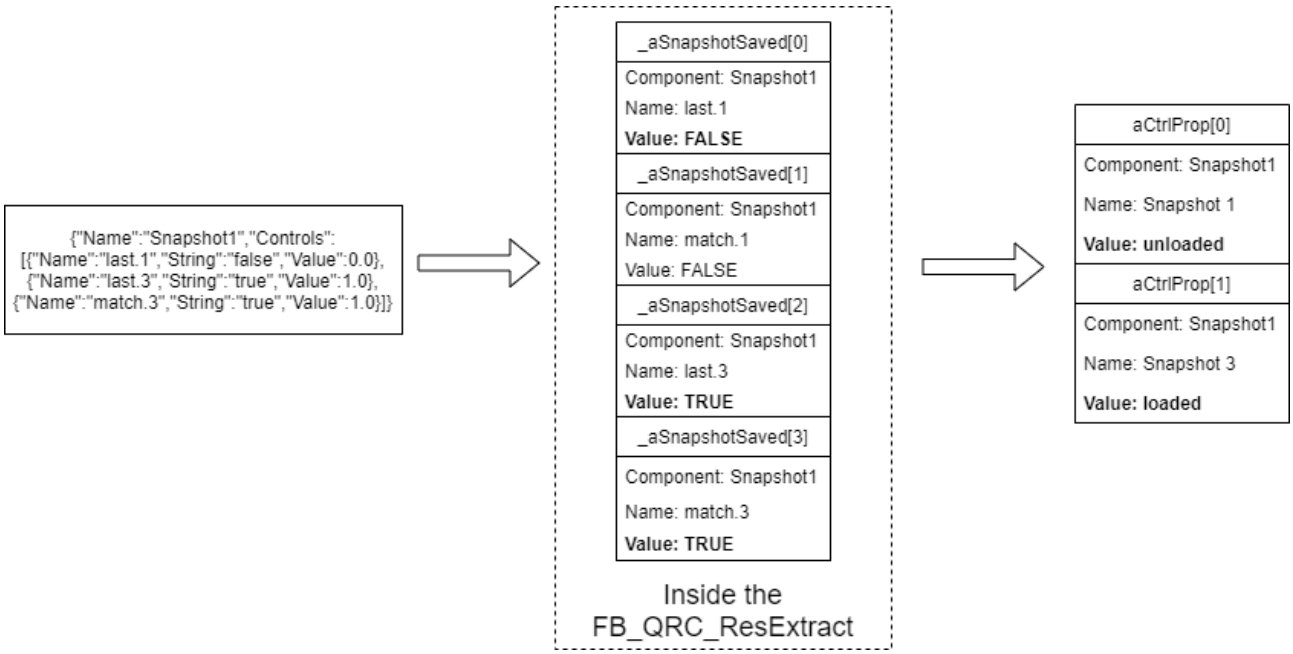
All related informations will be stored internally in an array. Snapshots' states are determined.

Step 2: In case some snapshot contained controls were changed within a polling cycle, a polling frame is arrived:



The property "`match.1`" will be updated in the internal array and the snapshot "Snapshot 1" changes its state from "loaded" to "changed". (`aCtrlProp[0]`)

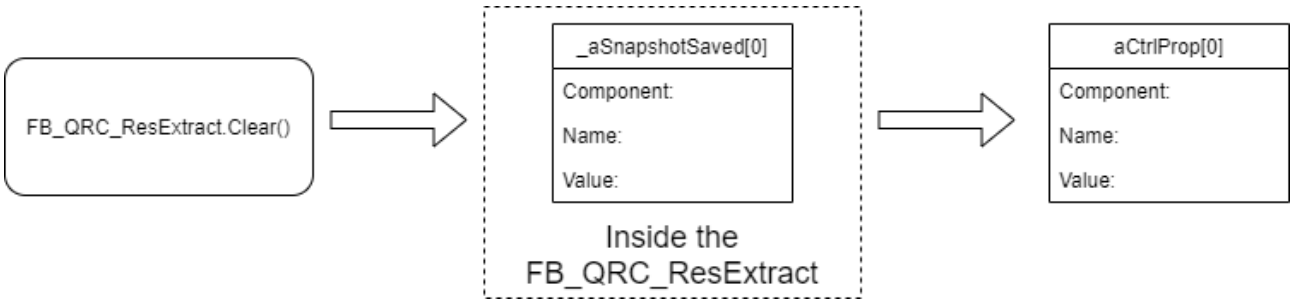
Step 3: Snapshot 3 is triggered.



The "Snapshot 3" was just triggered and the polling frame was received. Related control properties will be updated.

Step 4: Clear the internal array.

If users want to poll another snapshots' state and the stored properties are no longer useful, the method Clear [▶ 30] should be used to reset the internal array.

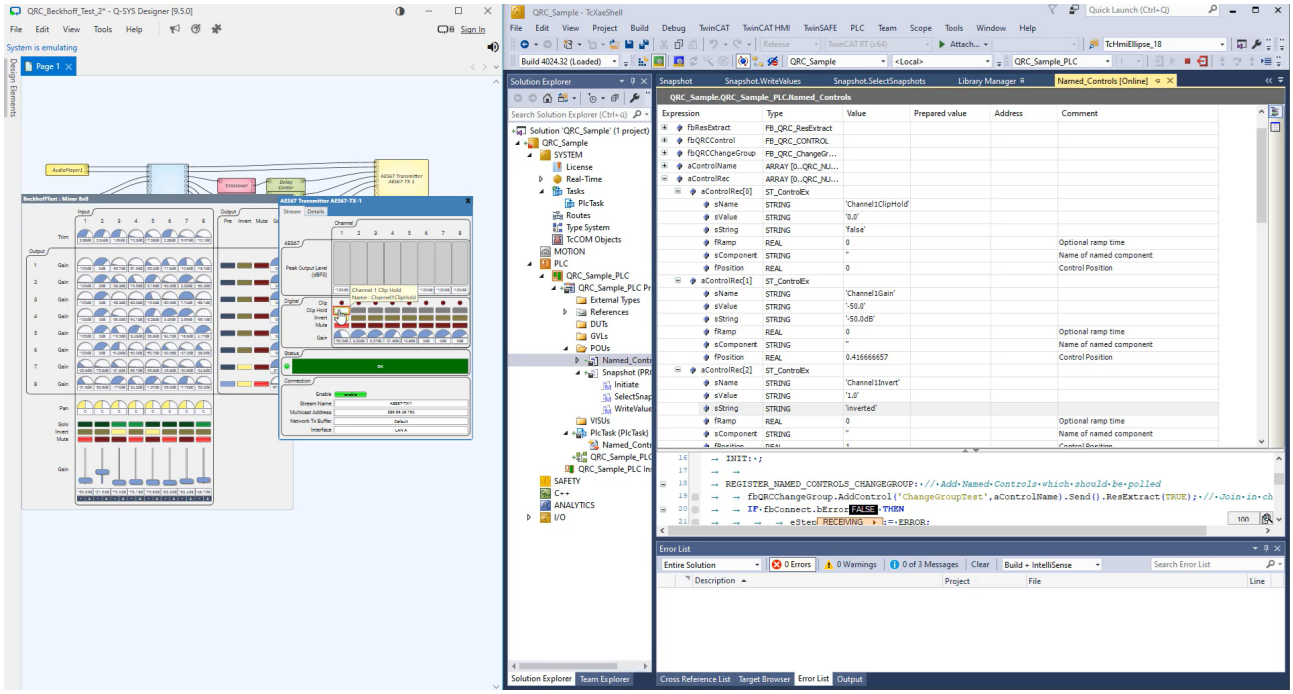


After the clear operation the internal array and the array `aCtrlProp` are both empty.

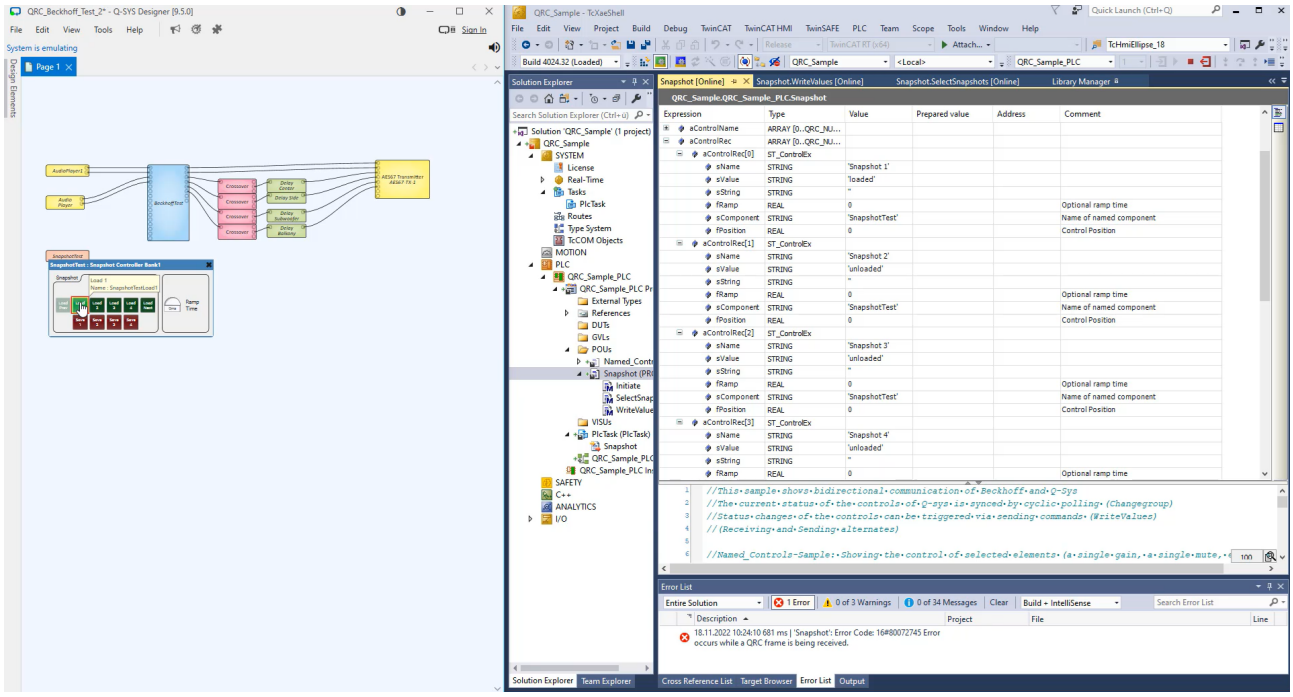
5 Example: AutoPolling and writing controls

The https://infosys.beckhoff.com/content/1033/TF6310_QRC/Resources/13204173963/.zip shows exemplarily the functionality of the QRC integration in TwinCAT. It consists of two TwinCAT programs and a Q-Sys design file. Additionally, two videos are included that show the basic functionality of both programs.

Named Controls can be selected from the Q-Sys Designer in the program "Named_Controls". The status of the controls is then get at any interval (AutoPoll) and values can be set for any controls.



In the "Snapshot" program you can select predefined snapshots from the Q-Sys Designer. The state of the snapshots is then get at any interval (AutoPoll) and the individual snapshots, snapshot 1-3 in this example, can be triggered.



6 Appendix

6.1 Error Codes

The following error codes can be returned. These error codes are defined by QSC.

Code(dec)	Description
-32700	Parse error. Invalid JSON was received by the server.
-32600	Invalid request. The JSON sent is not a valid Request object.
-32601	Method not found.
-32602	Invalid params.
-32603	Server error
2	Invalid Page Request ID.
3	Bad Page Request / could not create the request Page Request.
4	Missing file
5	Change Groups exhausted
6	Unknown change group
7	Unknown component name
8	Unknown control
9	Illgal mixer channel index
10	Logon required

You can also find related information in the "Error List" window.

6.2 Buffer size

During the sending process or the receiving process, a long TCP frame (length > `QRC_BUFFER_SIZE`) will be divided into more segments. After each receiving process the function block FB_Connect [▶ 13] will check received frame whether it is an independent QRC frame, or it is one segment of a long QRC frame. The function block will keep receiving until all segments are arrived or a receiving timeout occurs.

The size of the receiving buffer is 255 byte*`QRC_NUMBER_OF_CONTROL`. If the buffer gets overflowed, an error will occur and further error details can be found in the "Error List" window.

6.3 String function

The STRING functions (LEN, MID, LEFT, etc.) are only valid for normal string type (String length <= 255). For long string type (length > 255, in this project `sTxFrame` is a long string), memory functions (MEMSET, MEMCPY, MEMMOVE) can be used instead.

6.4 Easy way to find control name, component name and name of Snapshot Bank

Update:

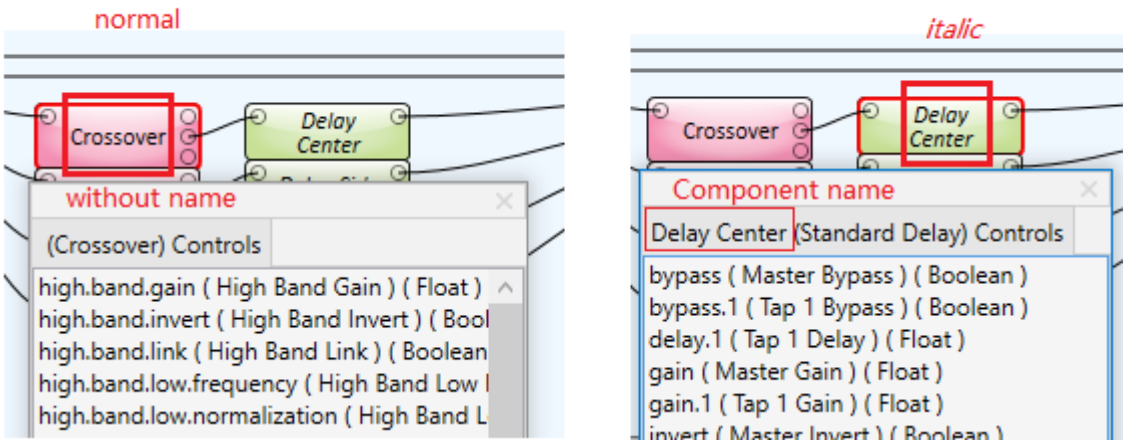
Since the version 1.1.0.0, name of a Snapshot Bank is needed when function block `FB_QRC_Snapshot` is used. Name of a Snapshot bank can be found and configured at the snapshot pane or the snapshot property window.

In contest to the TwinCAT program, it is also possible to instantiate controls and components without naming in Q-SYS Designer. However, control's names and component's names are the key to remote control. Each control or component cannot be accessed or controlled without a unique name. It is important to check whether each target control or component has a valid name or not before operation.

Therefore:

- For **controls**, the best way is to check whether its name has already been placed in the “Named Control” pane or not.
- For **components,** the easiest way is to check the font style on this component. Its font style is normal, means this component is not named yet. A named component’s text is in italic style.

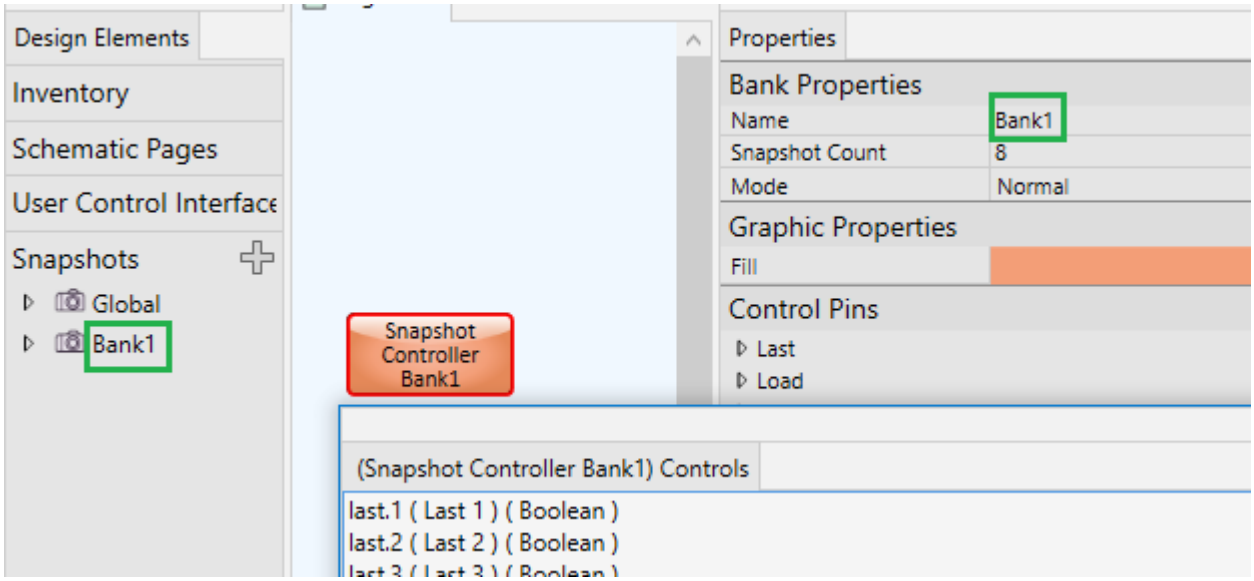
Here is an example for component:



(normal = “not named”, italic = named)

On the left side, the font of text "Crossover" is in normal type. It means "Crossover" is the component type and it is not named. At the right side, the font of text “*Delay Center*” is in italic style. The type of this control is "Standard Delay" and its name of is "Delay Center". It has a valid name.

Here is an example for snapshot bank:



(Green Rect = Name of a Snapshot Bank)

6.5 Control button "Load" of snapshot component

Update:

Since the version 1.1.0.0, you can use the method Load [▶ 39] and Save [▶ 39] of function block FB_QRC_Snapshot [▶ 38] to trigger / save snapshots without using the Named Control concept. Method GetSnapshotState [▶ 40] can be used as querying the snapshots' state manually, or the method AddSnapshotControl [▶ 31] of FB_QRC_ChangeGroup [▶ 28] can be used to join in a change group and then. The function block FB_QRC_ResExtract [▶ 16] is now supported to extract the response frame of snapshot. Read the section Snapshot state and related properties [▶ 52] for more information.

Foreword

There isn't any related information about controlling snapshot components in the QRC specification. The button "Load" (a control of snapshot component) is a "trigger" type. So there is no way of getting a status back, nor of adding them to a Change Group. The following solution is a functional workaround to get the feedback via the snapshot buttons' "color" property. Otherwise we can't ensure whether the "Load" process was executed successfully.

Using the "color" property it is possible to recognize status changes of buttons within a snapshot component. With the help of the "Control.Get [▶ 25]" command, the "color" property can be queried. Due to the behavior of "Save" buttons (No status change), they are excluded from this solution.

Table 1: Color property and its corresponding snapshot state

Color	State
'@7F19'	'unloaded'
'@7F7F'	'loaded'
'@7F4C'	'changed'

The 'changed' state means, relative controls have been changed after the snapshot was loaded. In this state, "Save" buttons are usable to overwrite a snapshot.

This logic has already been implemented in the function block FB_QRC_ResExtract [▶ 16].

1. This solution only works with the default button colors. DO NOT change the snapshot button color. Otherwise its status cannot be recognized by function block `FB_QRC_ResExtract`.
2. This method is specially developed for snapshot buttons, it doesn't work for other controls with trigger type.
3. The "Load Prev" and "Load Next" buttons are excluded from this solution because the "color" property of them can't be queried from Q-SYS device.

6.6 Snapshot state and related properties

The previous version of the QRC demo project, the "Color" property of the Load button was used to recognize the snapshot state. This is an unofficial "workaround" but it is still working well. Now, a new way has been implemented. The "Color" functionality still remaines.

Since the version 1.1.0.0, the function block FB_QRC_Snapshot [▶ 38] can now be used to load or save a snapshot directly, and to query multiple snapshots manually. With the help of method AddSnapshotControl [▶ 31] of the function block FB_QRC_ChangeGroup [▶ 28] multiple snapshots can be joined in a change group. Afterwards their states can be polled cyclically. The function block FB_QRC_ResExtract [▶ 16] has also been updated in order to extract the response frame of a snapshot.

A Snapshot Bank consists of a Snapshot Controller, and all the controls and components you add to it. This Snapshot Controller is also a component control and is called "snapshot component" in the following documentation for a better understanding.

Within a snapshot component, each snapshot has two related properties/component controls, which are listed below:

- **last.x**: It describes whether the snapshot is loaded or not.
- **match.x**: It describes whether Controls within the snapshot have been changed after this snapshot was loaded.

These two property names can be found using the menu "View Component Control Info..." in Q-SYS Designer. With the help of these two properties, the snapshot state can be determined.

"last" and "match" property value and their corresponding snapshot state are listed below.

	last = false	**last = true**
match = false	unloaded	changed
match = true	-	loaded

This logic has been implemented in the function block FB_QRC_ResExtract [▶ 16] and the method AddSnapshotControl [▶ 31] and GetSnapshotState [▶ 40].

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