Application note

Network scan

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC100106AN05EN | Revision 5 | English | 2017-01 | Released | Public

Table of contents

1 Introduction 3
1.1 About this document 3
1.2 List of revisions 3
1.3 Legal notes 4
1.3.1 Copyright 4
1.3.2 Important notes 4
1.3.3 Exclusion of liability 5
1.3.4 Export 5
2 Packet interface to scan the network 6
2.1 Network scan 6
2.2 Get device info 11
2.3 State machine of the network scan 14
3 Appendix 17
3.1 List of tables 17
3.2 List of figures 17
3.3 Contacts 18

1 Introduction

1.1 About this document

This document describes the application interface of the network scan functionality that is protocol stack independent and can be used by a host application of a master protocol stack. This includes the packets that need to be sent and received by the host application as well as the behavior of the network scan state machine.

1.2 List of revisions

Rev	Date	Name	Revision
5	2017-01-23	KM, HH	Revised

Table 1: List of revisions

1.3 Legal notes

1.3.1 Copyright

1.3.2 Important notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.3.3 Exclusion of liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.3.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different counters, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

2 Packet interface to scan the network

2.1 Network scan

This packet allows you to request a device list, i.e. a list of devices participating in the network at the current time. In order to request a device list, you do not have to specify any parameters. The confirmation packet contains the requested live list within the field `abDeviceList[]`. The length of the device list is protocol stack specific.

For each found device on the network a bit in the field `abDeviceList` is set to 1. The bit position inside the list (index) is further used as a unique handle for each found device and can be used to retrieve protocol stack specific device information as described in chapter *Get device info* on page 11.

The packet RCX_BUSSCAN_REQ can be used to perform three actions (`ulAction`):

- RCX_BUSSCAN_CMD_START

  This action starts the new network scan
- RCX_BUSSCAN_CMD_STATUS

  This action is used to poll the status/progress of a network scan
- RCX_BUSSCAN_CMD_ABORT (optionally)

  This action aborts a running network scan. The scan is set back into the initial state, results are flushed.

If `abDeviceList[]` holds two valid bytes in the confirmation packet (packet length `ulLen` = 10), the bits 0 to 7 of the first byte references device indexes 0 to 7. The second byte references device indexes 8 to 15. If the device list is longer, additional device indexes are referenced in ascending order.

Figure 1 shows how the host application should use the packet RCX_BUSSCAN_REQ in order to perform a network scan properly. Once the “Finish” state is reached, the packet RCX_DVICE_INFO_REQ can be used to obtain information about found devices.

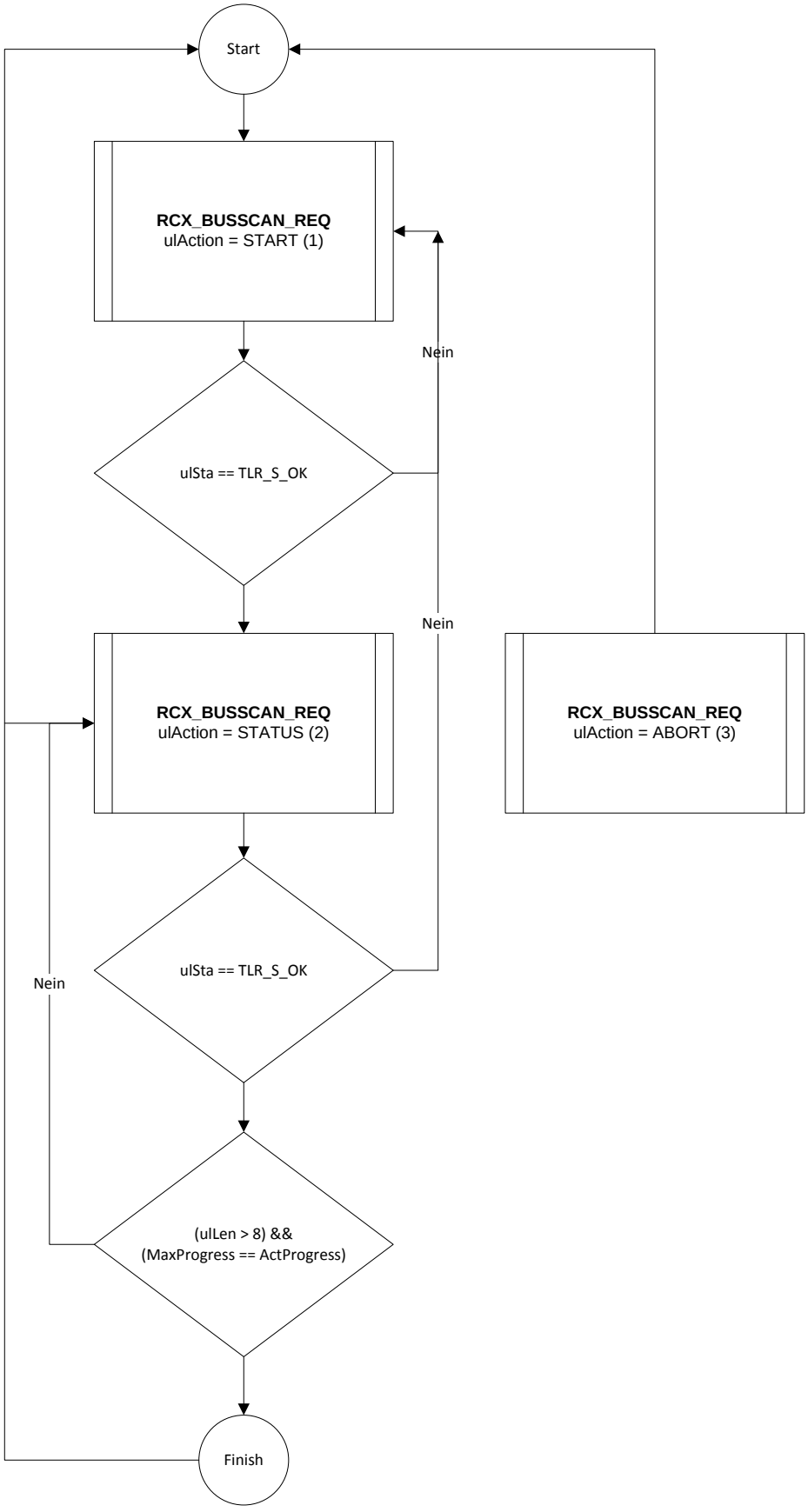

Figure 1: How to use the packet RCX_BUSSCAN_REQ

If the protocol stack receives a network scan action that it does not expect in its current state, the confirmation packet will contain the error code TLR_E_PACKET_OUT_OF_SEQ. This could for example happen if the action “Status” is requested although a network scan has not been started before.

Packet structure reference

```
#define RCX_BUSSCAN_CMD_START        0x01
#define RCX_BUSSCAN_CMD_STATUS       0x02
#define RCX_BUSSCAN_CMD_ABORT        0x03

#define RCX_BUSSCAN_REQ 0x2f22

typedef struct RCX_BUSSCAN_REQ_DATA_Ttag
{
  TLR_UINT32 ulAction;
} RCX_BUSSCAN_REQ_DATA_T;

typedef struct RCX_BUSSCAN_REQ_Ttag
{
  TLR_PACKET_HEADER_T    tHead;
  RCX_BUSSCAN_REQ_DATA_T tData;
} RCX_BUSSCAN_REQ_T;

#define RCX_BUSSCAN_REQ_SIZE     (sizeof(RCX_BUSSCAN_REQ_DATA_T))
```

Packet description

Structure RCX_BUSSCAN_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		Destination queue handle, unchanged
ulSrc	UINT32		Source queue handle, unchanged
ulDestId	UINT32		Destination end point identifier, unchanged
ulSrcId	UINT32		Source end point identifier, unchanged
ulLen	UINT32	4	Packet Data Length in bytes
ulId	UINT32	0 ... 2^{32}-1	Packet Identification as unique number generated by the Source Process of the Packet
ulSta	UINT32		Status/error code
ulCmd	UINT32	0x2F22	RCX_BUSSCAN_REQ - Command
ulExt	UINT32	0	Extension, unchanged
ulRout	UINT32	x	Routing, do not change
Structure RCX_BUSSCAN_REQ_DATA_T			
ulAction	UINT32	1-3	RCX_BUSSCAN_CMD_START (0x01) RCX_BUSSCAN_CMD_STATUS (0x02) RCX_BUSSCAN_CMD_ABORT (0x03)

Table 2: RCX_BUSSCAN_REQ – Network scan request

Packet structure reference

```
#define RCX_BUSSCAN_CNF 0x2f23

typedef struct RCX_BUSSCAN_CNF_DATA_Ttag
{
  TLR_UINT32 ulMaxProgress;
  TLR_UINT32 ulActProgress;
  TLR_UINT8  abDeviceList[4];
} RCX_BUSSCAN_CNF_DATA_T;

typedef struct RCX_BUSSCAN_CNF_Ttag
{
  TLR_PACKET_HEADER_T tHead;
  RCX_BUSSCAN_CNF_DATA_T tData;
} RCX_BUSSCAN_CNF_T;

#define RCX_BUSSCAN_CNF_SIZE     (sizeof(RCX_BUSSCAN_CNF_DATA_T) - 4)
```

Packet description

Structure RCX_BUSSCAN_CNF_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		Destination queue handle, unchanged
ulSrc	UINT32		Source queue handle, unchanged
ulDestId	UINT32		Destination end point identifier, unchanged
ulSrcId	UINT32		Source end point identifier, unchanged
ulLen	UINT32	8+n (if ok) 0 (otherwise)	Packet Data Length in bytes n = sizeof(abDeviceList), truncated after the byte with the last found slave
ulId	UINT32	0 ... 2^{32}-1	Packet Identification as unique number generated by the Source Process of the Packet
ulSta	UINT32		Status/error code
ulCmd	UINT32	0x2F23	RCX_BUSSCAN_CNF - Command
ulExt	UINT32	0	Extension, unchanged
ulRout	UINT32	x	Routing, do not change
Structure RCX_BUSSCAN_CNF_DATA_T			
ulMaxProgress	UINT32		Maximum progress
ulActProgress	UINT32		Actual progress
abDeviceList	UINT8		Bit list of found devices. The bit list is defined as abDeviceList[4], which means that there can be reported up to 32 devices. However, depending on the protocol stack there could also be more valid bytes. Therefore, always the ulLen field of the packet header must be evaluated first.

Table 3: RCX_BUSSCAN_CNF – Confirmation of network scan request

Error Codes of the RCX_BUSSCAN_CNF

Hexadecimal Value	Definition / Description
0x00000000	TLR_S_OK Status ok
0xC0000007	TLR_E_INVALID_PACKET_LEN Invalid packet length.
0xC000001A	TLR_E_REQUEST_RUNNING A new request was received, but one request is still running.
0xC000000F	TLR_E_PACKET_OUT_OF_SEQ A packet index has been not in the expected sequence
0xC0000009	TLR_E_INVALID_PARAMETER Invalid Parameter in Packet found

Table 4: Error codes of the RCX_BUSSCAN_CNF

2.2 Get device info

This packet can be used to request information of a device that has been found during previously performed network scan.

The device information within the confirmation packet is protocol stack specific. Each protocol stack defines its own device information structure. In order to be able to distinguish different protocol stack structures, each protocol stack specific structure is assigned a unique structure ID.

Packet structure reference

```
#define RCX_GET_DEVICE_INFO_REQ 0x2f24

typedef struct RCX_GET_DEVICE_INFO_REQ_DATA_Ttag
{
  TLR_UINT32 ulDeviceIdx;
} RCX_GET_DEVICE_INFO_REQ_DATA_T;

typedef struct RCX_GET_DEVICE_INFO_REQ_Ttag
{
  TLR_PACKET_HEADER_T            tHead;
  RCX_GET_DEVICE_INFO_REQ_DATA_T tData;
} RCX_GET_DEVICE_INFO_REQ_T;

#define RCX_GET_DEVICE_INFO_REQ_SIZE     (sizeof(RCX_GET_DEVICE_INFO_REQ_DATA_T))
```

Packet description

Structure RCX_GET_DEVICE_INFO_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		Destination queue handle, unchanged
ulSrc	UINT32		Source queue handle, unchanged
ulDestId	UINT32		Destination end point identifier, unchanged
ulSrcId	UINT32		Source end point identifier, unchanged
ulLen	UINT32	4	Packet Data Length in bytes
ulId	UINT32	0 ... 2^{32}-1	Packet Identification as unique number generated by the Source Process of the Packet
ulSta	UINT32		Status/error code
ulCmd	UINT32	0x2F24	RCX_GET_DEVICE_INFO_REQ - Command
ulExt	UINT32	0	Extension, unchanged
ulRout	UINT32	x	Routing, do not change
Structure RCX_GET_DEVICE_INFO_REQ_DATA_T			
ulDeviceIdx	UINT32		Device index

Table 5: RCX_GET_DEVICE_INFO_REQ – Device info command

Packet structure reference

```
#define RCX_GET_DEVICE_INFO_CNF 0x2f25

typedef struct RCX_GET_DEVICE_INFO_CNF_DATA_Ttag
{
  TLR_UINT32 ulDeviceIdx;
  TLR_UINT32 ulStructId;
  /* TLR_UINT8  tStruct; Protocol stack specific structure */
} RCX_GET_DEVICE_INFO_CNF_DATA_T;

typedef struct RCX_GET_DEVICE_INFO_CNF_Ttag
{
  TLR_PACKET_HEADER_T            tHead;
  RCX_GET_DEVICE_INFO_CNF_DATA_T tData ;
} RCX_GET_DEVICE_INFO_CNF_T;

#define RCX_GET_DEVICE_INFO_CNF_SIZE     (sizeof(RCX_GET_DEVICE_INFO_CNF_DATA_T) - 1)
```

Packet description

Structure RCX_GET_DEVICE_INFO_CNF_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		Destination queue handle, unchanged
ulSrc	UINT32		Source queue handle, unchanged
ulDestId	UINT32		Destination end point identifier, unchanged
ulSrcId	UINT32		Source end point identifier, unchanged
ulLen	UINT32	8+n (if ok) 0 (otherwise)	Packet Data Length in bytes n = sizeof(tStruct)
ulId	UINT32	0 ... 2^{32}-1	Packet Identification as unique number generated by the Source Process of the Packet
ulSta	UINT32		Status/error code
ulCmd	UINT32	0x2F25	RCX_GET_DEVICE_INFO_CNF - Command
ulExt	UINT32	0	Extension, unchanged
ulRout	UINT32	x	Routing, do not change
Structure RCX_GET_DEVICE_INFO_CNF_DATA_T			
ulDeviceIdx	UINT32		Device Index
ulStructId	UINT32		Structure ID
tStruct	struct		Communication-system specific structure, see below

Table 6: RCX_GET_DEVICE_INFO_CNF – Confirmation of device info request

Error codes of the RCX_GET_DEVICE_INFO_CNF

Hexadecimal value	**Definition / Description**
0x00000000	TLR_S_OK Status ok
0xC0000007	TLR_E_INVALID_PACKET_LEN Invalid packet length.
0xC000000F	TLR_E_PACKET_OUT_OF_SEQ A packet index has been not in the expected sequence (Network Scan not started before or it was aborted)
0xC0000009	TLR_E_INVALID_PARAMETER Invalid Parameter in Packet found

Table 7: Error codes of the RCX_GET_DEVICE_INFO_CNF

Example for tStruct result

This structure depends on the used protocol stack. Therefore, have a look into the appropriate protocol stack API manual.

The following illustrates an example for the EtherNet/IP master:

The public file EtherNetIP.c holds the following definition:

```
static const UINT16 ausSI_EIP_DEVICE_INFO_T[]={
  1343, 0,
  1, 19, 0,
  2, 18, 0,
  3, 18, 0,
  4, 18, 0,
  5, 17, 0,
  6, 17, 0,
  7, 18, 0,
  8, 19, 0,
  9, 100, 33,
  10, 17, 0,
};
```

The “magic number” 1343 is the value which is reported in ulStructId in the packet confirmation.

The public header file EtherNetIP.h provides the actual data structure that is behind the magic number:

```
typedef __PACKED_PRE struct __PACKED_POST EIP_DEVICE_INFO_Ttag {
  UINT32 ulIpAddress;
  UINT16 usVendorId;
  UINT16 usDeviceType;
  UINT16 usProductCode;
  UINT8  bMajorRevision;
  UINT8  bMinorRevision;
  UINT16 usStatus;
  UINT32 ulSerialNumber;
  STRING szProductName[33];
  UINT8  bState;
} EIP_DEVICE_INFO_T;
```


2.3 State machine of the network scan

Independent of the underlying protocol stack, the behavior of the network scan implementation is shown by the following state diagram:

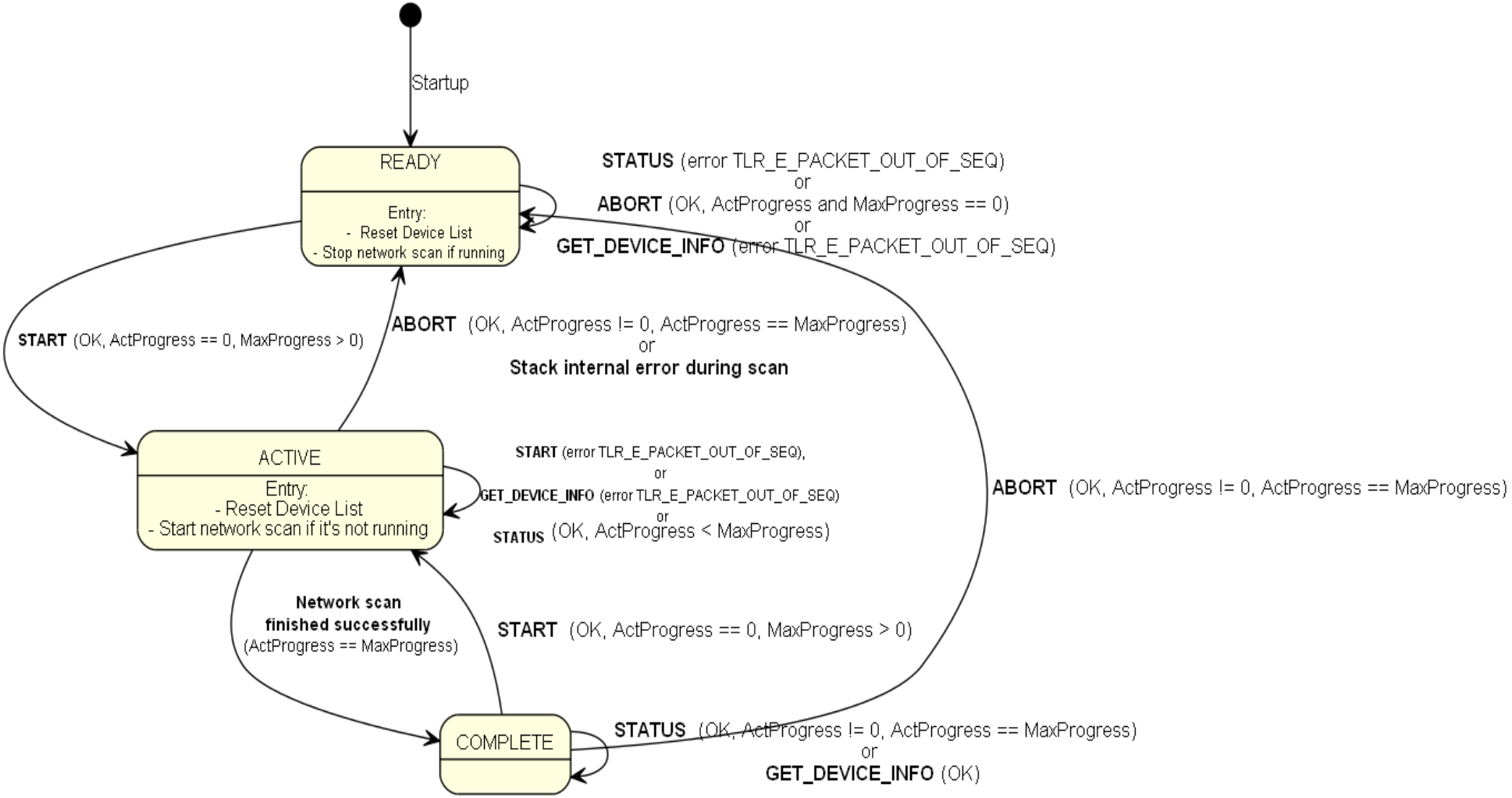

Figure 2: State machine of the network scan implementation within the protocol stack

States of the state machine

1. READY

   No network scan is running. The device list is cleared.

   In this state the following commands are allowed. All other commands will be rejected with the error illustrated in the state machine diagram.

   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_START)
   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_ABORT)

2. ACTIVE

   A network scan is running. The current status can be request with a status command.

   In this state the following commands are allowed. All other commands will be rejected with the error illustrated in the state machine diagram.

   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_STATUS)
   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_ABORT)

3. COMPLETE

   A network scan finished successfully. The current status can be request with a status command and the device information can be requested with the packet RCX_GET_DEVICE_INFO_REQ. A new network scan can be started.

   In this state the following commands are allowed. All other commands will be rejected with the error illustrated in the state machine diagram.

   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_START)
   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_STATUS)
   - RCX_BUSSCAN_REQ (ulAction = RCX_BUSSCAN_CMD_ABORT)
   - RCX_GET_DEVICE_INFO_REQ

The following diagram shows how the packets are exchanged between the host application and the protocol stack when a network scan is executed successfully.

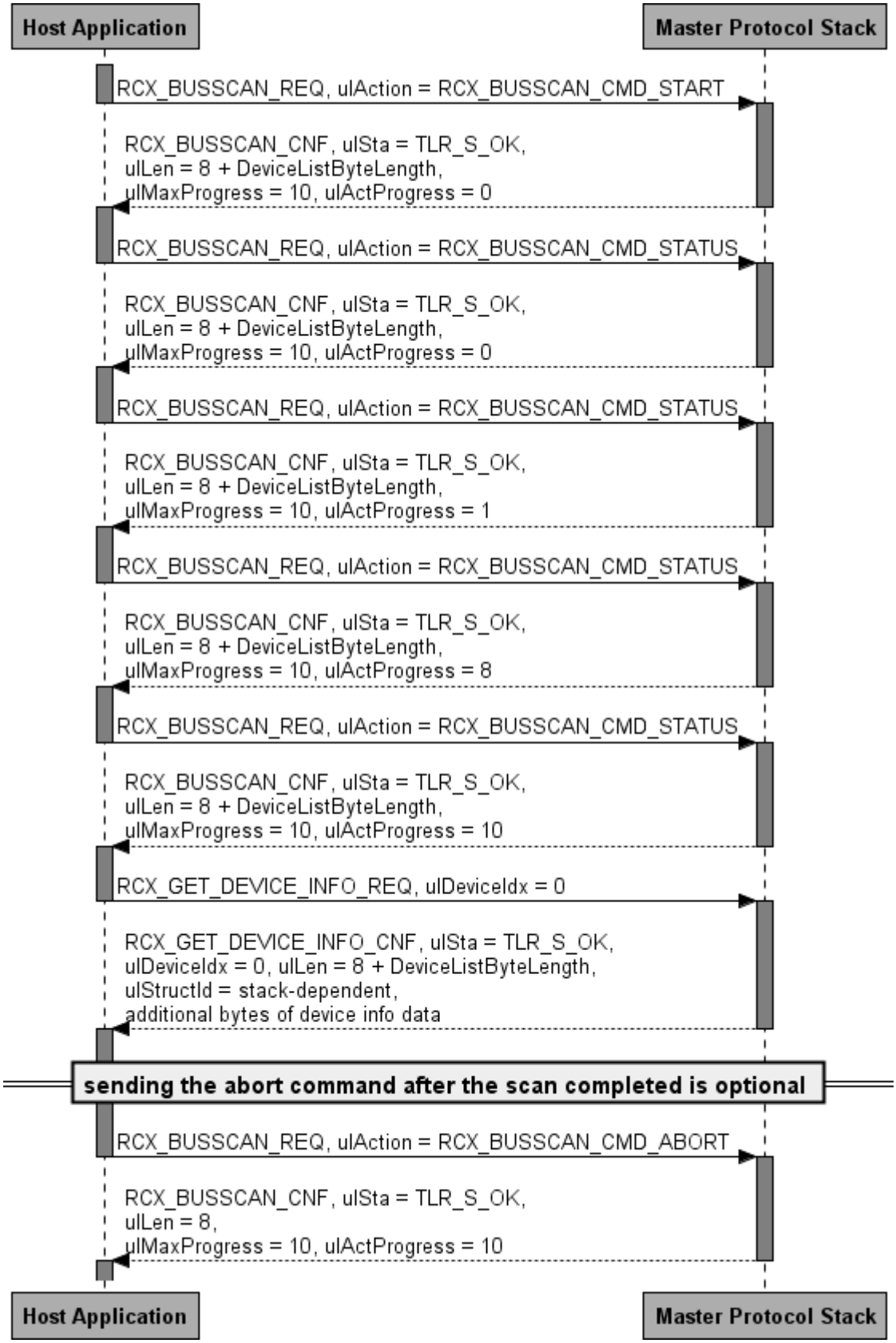

Figure 3: Successful network scan sequence

3 Appendix

3.1 List of tables

Table 1: List of revisions 3
Table 2: RCX_BUSSCAN_REQ – Network scan request 8
Table 3: RCX_BUSSCAN_CNF – Confirmation of network scan request 9
Table 4: Error codes of the RCX_BUSSCAN_CNF 10
Table 5: RCX_GET_DEVICE_INFO_REQ – Device info command 11
Table 6: RCX_GET_DEVICE_INFO_CNF – Confirmation of device info request 12
Table 7: Error codes of the RCX_GET_DEVICE_INFO_CNF 12

3.2 List of figures

Figure 1: How to use the packet RCX_BUSSCAN_REQ 7
Figure 2: State machine of the network scan implementation within the protocol stack 14
Figure 3: Successful network scan sequence 16

3.3 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-Mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com

