Reading and Writing RFID Data with SIMATIC S7-300/400 and SIMATIC RF670R

SIMATIC RF670R

Application Description • April 2013
Caution
The functions and solutions described in this entry predominantly confine themselves to the realization of the automation task. Please also take into account that corresponding protective measures have to be taken in the context of Industrial Security when connecting your equipment to other parts of the plant, the enterprise network or the Internet. For more information, please refer to Entry ID 50203404.

SIMATIC
Reading from and Writing to Transponders with RF670R

Application Description
Warranty and Liability

Note

The application examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The application examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are correctly used. These application examples do not relieve you of the responsibility of safely and professionally using, installing, operating and servicing equipment. When using these application examples, you recognize that Siemens cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these application examples at any time without prior notice. If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – then the contents of the other documents have priority.

We do not accept any liability for the information contained in this document. Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc. described in this application example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above provisions do not imply a change in the burden of proof to your detriment.

It is not permissible to transfer or copy these application examples or excerpts of them without first having prior authorization from Siemens Industry Sector in writing.
# Table of Contents

Warranty and Liability ........................................................................................................... 4

Table of Contents .................................................................................................................. 5

1 Problem .............................................................................................................................. 7

2 Solution ................................................................................................................................ 8

  2.1 Overview of the overall solution ....................................................................................... 8
  2.2 Description of the core functionality .................................................................................. 9
  2.3 Hardware and software components used ........................................................................ 11
  2.4 Performance data ............................................................................................................ 13

3 Functional Mechanisms of this Application ....................................................................... 16

  3.1 Program overview .......................................................................................................... 16
  3.1.1 FB RF670R_RW parameter .................................................................................... 17
  3.1.2 RF_VAR structure .................................................................................................... 18
  3.1.3 Call example: FB RF670R_RW (FB670) in FB CALL_RF670R (FB1) ....................... 21
  3.1.4 State diagram of FB RF670R_RW (FB670) ............................................................. 22
  3.1.5 Function chart .......................................................................................................... 24
  3.2 Explanation of the functions implemented in FB RF670R_RW ....................................... 25
      3.2.1 “Connect or disconnect” ....................................................................................... 25
      3.2.2 “Read or write transponder IDs” ................................................................. 29
      3.2.3 “Read RFID data from or write RFID data to transponder” .......................... 35
  3.3 Error and status display ................................................................................................. 42
      3.3.1 Error messages of FB RF670R_RW (FB670) .................................................... 42
      3.3.2 Error messages of the RF670R reader ............................................................... 43
      3.3.3 Error messages of communication blocks ......................................................... 43

4 Installation and Commissioning ......................................................................................... 44

  4.1 Hardware configuration ................................................................................................. 44
  4.2 Hardware installation: S7 station .................................................................................... 45
  4.3 Hardware installation: RF670R reader .......................................................................... 45
  4.4 Installation of the standard software ............................................................................. 45
  4.5 Installing the STEP 7 project ......................................................................................... 46
  4.6 Setting the PG/PC interface .......................................................................................... 46
  4.7 Configuring the PG/PC ................................................................................................. 47
  4.8 Configuring the S7 station ............................................................................................ 48
  4.9 Configuring the RF670R reader with RF-MANAGER Basic 2010 .............................. 51
  4.10 Downloading the STEP 7 project ............................................................................... 53
  4.11 Orientation of the transponders with respect to the antenna ...................................... 53

5 Operation of the Application ............................................................................................. 55

  5.1 Overview ......................................................................................................................... 55
  5.2 Connecting to the RF670R ............................................................................................ 56
  5.3 Reading the transponder IDs ......................................................................................... 57
  5.4 Writing the transponder ID ........................................................................................... 58
Table of Contents

5.5 Reading RFID data from a transponder .......................................................... 60
5.6 Writing RFID data to a transponder .............................................................. 61
5.7 Disconnecting ................................................................................................. 62

6 Further Information .......................................................................................... 63
   6.1 Modifications to the program ........................................................................ 63
   6.2 Auxiliary blocks for converting a SIMATIC variable type for the RFID reader ................................................................. 66
       6.2.1 FB INT_TO_HEX and FB HEX_TO_INT parameters ........................................ 66
       6.2.2 Call of FB INT_TO_HEX and FB HEX_TO_INT in OB1 ................................. 68

7 References ........................................................................................................... 69
   7.1 References ...................................................................................................... 69
   7.2 Internet links .................................................................................................. 69

8 History ................................................................................................................ 70
1 Problem

Introduction
For incoming goods, stock keeping, production logistics and distribution, RFID (Radio Frequency Identification) provides complete tracking and documentation of all delivered, stored and sent goods. For this purpose, a small data medium – referred to as a transponder or tag – that stores all essential information is attached to each product, package or pallet.
A read/write device is used to read from and write to the transponder. To ensure complete tracking and documentation or to benefit from the use of RFID technology even during production, the connection to automation systems is required in many cases.

Overview of the automation problem
The figure below provides an overview of the automation problem.

Figure 1-1

The high-performance RF670R reader from the RFID UHF system with
- integrated processing logic that allows comprehensive filter functions
- reliable identification at large intervals
- reliable detection of fast-moving transponders and
- an Ethernet interface (XML Protocol via TCP/IP) for easy connection to the PC or IT level
is to be connected to an S7-300/400 CPU with PROFINET interface and implement the reading from and writing to transponders.
2 Solution

2.1 Overview of the overall solution

Introduction

This application example shows you

- how to parameterize the RF670R reader using RF-MANAGER Basic 2010 regarding the reading and writing of data
- how to program a block in your S7-300/400 CPU with PROFINET interface to implement the following core functions:
  - Connect to/disconnect from reader
  - Read/write transponder ID
  - Read/write RFID data
- how to format SIMATIC S7 variable types for the RF670R.

The complete functionality is encapsulated in an open SCL block and can be used as a basis for your own developments such as other RF670R functions.

Diagrammatic representation

The diagrammatic representation below shows the most important components of the solution with an S7-300 CPU with PROFINET interface. Alternatively, the solution can also be implemented with another S7-300/400 CPU with PROFINET interface that supports TCP (see [3]).

Figure 2-1
2 Solution

2.2 Description of the core functionality

Components included

Table 2-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PROFINET S7 CPU</td>
<td>An S7-300 CPU, S7-400 CPU</td>
</tr>
<tr>
<td>2</td>
<td>SIMATIC RF670R</td>
<td>Reader</td>
</tr>
<tr>
<td>3</td>
<td>SIMATIC RF620A</td>
<td>Antenna</td>
</tr>
<tr>
<td>4</td>
<td>SIMATIC RF640T</td>
<td>Transponder</td>
</tr>
<tr>
<td>5</td>
<td>Wide-range power supply unit with Euro-plug</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Communication block for the reader</td>
<td>FB 670</td>
</tr>
</tbody>
</table>

Scope

This application does not include the basics of
- SIMATIC RF600. For more information, refer to document \4\ in References.
- SIMATIC RF670R. For more information, refer to document \5\ in References.
- the LAD/ FBD/ STL/ SCL programming languages.

Basic knowledge of these topics is required.

Validity of application V1.1

- All PROFINET S7-300/400 CPUs from the SIMATIC product range that support TCP (see \3\)
- STEP 7 V5.5
- SIMATIC RF670R, RF640R

Note

If you use an RF640R reader, you have to change in the FB RF670R_RW (FB670) the Reader-type in hostGreetings Message.

```
179  GREETING_STR1 := '</id><hostGreetings><readerType>SIMATIC_RF6640R<read
180  GREETING_STR2 := '</readerType><supportedVersions><versionV1.0<versio
```

2.2 Description of the core functionality

In this example, the following functions are implemented with a user block programmed in SCL: FB RF_670R_RW (FB 670):
- Establish or terminate the connection between a PROFINET S7-300/400 station and the RF670R RFID reader
- Read transponder ID
- Write transponder ID
- Read RFID data from transponder
- Write RFID data to transponder
2 Solution
2.2 Description of the core functionality

“Connect or disconnect”
Figure 2-2

Via the “CONNECT” input, the TCP/IP connection to the RF670 reader is established and the hostGreetings message is sent.
Via the “DISCONNECT” input, the hostGoodbye message is sent to the RF670 reader and the TCP/IP connection is terminated.

“Read or write transponder IDs”
Figure 2-3

Via the “READ_TAG_ID” input, all transponder IDs of the transponders (TAGs) detected in the field of the reader are read in.
Via the “WRITE_TAG_ID” input, a new transponder ID is written to the transponder detected first by the reader.
“Read RFID data from or write RFID data to transponder”

Figure 2-4

Via the READ_MEMORY input, the RFID data is read out of the transponder memory.
Via the WRITE_MEMORY input, the RFID data is written to the transponder memory.

Advantages of this solution

The code of the FB “RF670R_RW” block (FB670)
- already includes the above-described functions on a fully implemented basis
- can be easily adapted for extensions such as other RF670R functions.

2.3 Hardware and software components used

The application was created with the following components:

Hardware components

Table 2-2

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>Order no.</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS307 10A</td>
<td>1</td>
<td>6ES7307-1KA01-0AA0</td>
<td></td>
</tr>
<tr>
<td>CPU 315-2 PN/DP</td>
<td>1</td>
<td>6ES7315-2EH14-0AB0</td>
<td>Alternatively, another CPU with PROFINET interface that supports TCP can also be used.</td>
</tr>
<tr>
<td>SIMATIC RF670R (reader)</td>
<td>1</td>
<td>6GT2811-0AB00-0AA0 (EU) 6GT2811-0AB00-1AA0 (USA) 6GT2811-0AB00-2AA0 (CHINA)</td>
<td>Alternatively, RF640R can also be used.</td>
</tr>
<tr>
<td>Connecting cable for power units</td>
<td>1</td>
<td>6GT2891-0NH50</td>
<td>5 m</td>
</tr>
<tr>
<td>SIMATIC RF640T</td>
<td>n</td>
<td>6GT2810-0DC00</td>
<td></td>
</tr>
</tbody>
</table>
2 Solution

2.3 Hardware and software components used

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty</th>
<th>Order no.</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>(transponder)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMATIC RF620A (antenna)</td>
<td>1</td>
<td>6GT2812-1EA00</td>
<td>Please note: ETSI frequency. Up to four antennas per RF670R</td>
</tr>
<tr>
<td>Antenna cable</td>
<td>1</td>
<td>6GT2815-0BH30</td>
<td>3 m</td>
</tr>
<tr>
<td>Wide-range power supply unit</td>
<td>1</td>
<td>6GT2898-0AA00 (EU)</td>
<td>6GT2898-0AA10 (UK) 6GT2898-0AA20 (US)</td>
</tr>
</tbody>
</table>

Standard software components

Table 2-3

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty</th>
<th>Order no.</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 7 V5.5</td>
<td>1</td>
<td>6ES7810-4CC08-0YA5</td>
<td></td>
</tr>
<tr>
<td>S7-SCL V5.3+SP5</td>
<td>1</td>
<td>6ES7811-1CC05-0YA5</td>
<td></td>
</tr>
<tr>
<td>SIMATIC RF-MANAGER Basic 2010, V2 or V3</td>
<td>1</td>
<td></td>
<td>Included in the scope of delivery of the RF670R</td>
</tr>
</tbody>
</table>

Sample files and projects

The following list contains all files and projects that are used in this example.

Table 2-4

<table>
<thead>
<tr>
<th>Component</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>23626344_RF670R_CODE_V11.zip</td>
<td>This zip file contains the STEP 7 project.</td>
</tr>
<tr>
<td>23626344_Application_RF670R_DOKU_V11_en.pdf</td>
<td>This document.</td>
</tr>
</tbody>
</table>
2.4 Performance data

The following section gives you an overview of the size of the blocks of the STEP 7 project in the main memory.

Blocks used and resource requirements

The size of all program blocks in the main memory is 36,342 Bytes. They are composed as follows.

Table 2-5

<table>
<thead>
<tr>
<th>Block</th>
<th>Symbol</th>
<th>Description</th>
<th>Size in main memory (bytes)</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB670</td>
<td>RF670R_RW</td>
<td>Function block for reading from and writing to transponders</td>
<td>23442</td>
<td>In-house development</td>
</tr>
<tr>
<td>DB670</td>
<td>IDB_RF670R_RW</td>
<td>Instance DB FB670</td>
<td>3442</td>
<td></td>
</tr>
<tr>
<td>DB1</td>
<td>RF_PARAM</td>
<td>Global data block for parameterizing the RF670R function block (FB670)</td>
<td>644</td>
<td></td>
</tr>
<tr>
<td>FB1</td>
<td>CALL_RF670R</td>
<td>Call of the FB670 function block with individual logic</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>DB3</td>
<td>IDB_CALL_RF670R</td>
<td>Instance DB FB1</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>FB2</td>
<td>INT_TO_HEX</td>
<td>Function block for converting INT to HEX format</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>DB4</td>
<td>IDB_INT_HEX</td>
<td>Instance DB FB2</td>
<td>448</td>
<td></td>
</tr>
<tr>
<td>FB3</td>
<td>HEX_TO_INT</td>
<td>Function block for converting HEX format to INT</td>
<td>2120</td>
<td></td>
</tr>
<tr>
<td>DB5</td>
<td>IDB_HEX_INT</td>
<td>Instance DB FB3</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td>In_Out_Data</td>
<td>Global data block for the converted data</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>UDT 670</td>
<td>RF_VAR</td>
<td>Data structure that contains the parameters necessary for RF670R_RW (FB670)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>UDT 65</td>
<td>TCON_PAR</td>
<td>Data structure that contains the parameters necessary to establish the connection</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>FB 65</td>
<td>TCON</td>
<td>Block for setting up and establishing the communication connection</td>
<td>1018</td>
<td>Standard Library/ communication blocks</td>
</tr>
<tr>
<td>FB 63</td>
<td>TSEND</td>
<td>TSEND (FB 63) sends data to the addressed remote partner via TCP/IP</td>
<td>416</td>
<td></td>
</tr>
<tr>
<td>FB 64</td>
<td>TRCV</td>
<td>TRCV (FB 64) receives data via TCP/IP</td>
<td>472</td>
<td></td>
</tr>
<tr>
<td>FB 66</td>
<td>TDISCON</td>
<td>This block disconnects a communication connection to a communication partner</td>
<td>230</td>
<td></td>
</tr>
</tbody>
</table>
## 2 Solution

### 2.4 Performance data

<table>
<thead>
<tr>
<th>Block</th>
<th>Symbol</th>
<th>Description</th>
<th>Size in main memory (bytes)</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC 2</td>
<td>CONCAT</td>
<td>This function combines two STRING variables in one string</td>
<td>358</td>
<td>Standard Library/IEC function blocks</td>
</tr>
<tr>
<td>FC 4</td>
<td>DELETE</td>
<td>This function deletes characters in a string</td>
<td>414</td>
<td></td>
</tr>
<tr>
<td>FC10</td>
<td>EQ_STRING</td>
<td>This function compares the contents of two strings</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>FC 16</td>
<td>I_STRING</td>
<td>The FC 16 function converts a variable in INT format to a string</td>
<td>264</td>
<td></td>
</tr>
<tr>
<td>FC 21</td>
<td>LEN</td>
<td>The FC 21 function outputs the current length of a string (number of valid characters) as a return value</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>FC 38</td>
<td>STRING_I</td>
<td>The FC 38 function converts a string to a variable in INT format</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>SFB4</td>
<td>TON</td>
<td>Generates a switch-on delay</td>
<td>-</td>
<td>Standard Library/system function blocks</td>
</tr>
<tr>
<td>SFC20</td>
<td>BLKMOV</td>
<td>Copies the contents of a memory area</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SFC21</td>
<td>FILL</td>
<td>Pre-fills a memory area (destination field) with the contents of another memory area (source field)</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Standard Library/IEC function blocks**
- SFB4 TON
- SFC20 BLKMOV
- SFC21 FILL
2.4 Performance data

Local data memory requirements

Table 2-6

<table>
<thead>
<tr>
<th>Block</th>
<th>Symbol</th>
<th>Local data</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB670</td>
<td>RF670R_RW</td>
<td>148 bytes</td>
</tr>
<tr>
<td>FB1</td>
<td>CALL_RF670R</td>
<td>6 bytes</td>
</tr>
<tr>
<td>FB2</td>
<td>INT_TO_HEX</td>
<td>48 bytes</td>
</tr>
<tr>
<td>FB3</td>
<td>HEX_TO_INT</td>
<td>22 bytes</td>
</tr>
<tr>
<td>FB 65</td>
<td>TCON</td>
<td>28 bytes</td>
</tr>
<tr>
<td>FB 63</td>
<td>TSEND</td>
<td>24 bytes</td>
</tr>
<tr>
<td>FB 64</td>
<td>TRCV</td>
<td>24 bytes</td>
</tr>
<tr>
<td>FB 66</td>
<td>TDISCON</td>
<td>12 bytes</td>
</tr>
</tbody>
</table>

Watchdog timer

A watchdog timer monitors connection establishment to the RF670R reader. It is preset to 5 s.

Note

If you want to change the time, you can enter the value directly in the instance data block of FB RF670R_RW (FB670).

Application software

The following table shows the measured runtimes of the functions in the S7-300 CPU from the test setup, the size of the receive buffer and the maximum number of read transponder IDs.

Table 2-7

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect runtime (CONNECT)</td>
<td>3 ms</td>
</tr>
<tr>
<td>Read transponder IDs runtime (READ_TAG_IDs)</td>
<td>10 ms</td>
</tr>
<tr>
<td>Write transponder ID runtime (WRITE_TAG_ID)</td>
<td>3 ms</td>
</tr>
<tr>
<td>Read RFID data runtime (READ_MEMORY)</td>
<td>8 ms</td>
</tr>
<tr>
<td>Write RFID data runtime (WRITE_MEMORY)</td>
<td>5 ms</td>
</tr>
<tr>
<td>Disconnect runtime (DISCONNECT)</td>
<td>3 ms</td>
</tr>
<tr>
<td>Maximum runtime of the sample program</td>
<td>12 ms</td>
</tr>
<tr>
<td>Receive buffer size</td>
<td>1500 bytes</td>
</tr>
<tr>
<td>Maximum number of read transponder IDs</td>
<td>5 (*1)</td>
</tr>
</tbody>
</table>

(*1): The number of stored transponder IDs depends on the field length of REC_DATA in DB RF_PARAMETER. If you want to read more transponder IDs, you have to adjust the receive buffer length (see Chapter 6.1).
3 Functional Mechanisms of this Application

3.1 Program overview

Introduction

This chapter provides a detailed description of the FB RF670R_RW user block (FB670) in terms of the internal functional sequences and programming.

Diagrammatic representation

The figure below shows the program structure of the entire STEP 7 project.

Figure 3-1
3 Functional Mechanisms of this Application

3.1 Program overview

3.1.1 FB RF670R_RW parameter

The following figure and table show the call interface of the FB RF670R_RW user block (FB670).

Figure 3-2

Table 3-1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>BOOL</td>
<td>Enable input. Relevant only in FBD and LAD representation.</td>
</tr>
<tr>
<td>CONNECT</td>
<td>BOOL</td>
<td>• Activates the connection establishment routine to the reader</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
<tr>
<td>READ_TAG_ID</td>
<td>BOOL</td>
<td>• Activates the routine for reading the transponder IDs of the tags in the field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
<tr>
<td>WRITE_TAG_ID</td>
<td>BOOL</td>
<td>• Activates the routine for writing the transponder ID of the transponder in the field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
<tr>
<td>READ_MEMORY</td>
<td>BOOL</td>
<td>• Activates the routine for reading out the RFID data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
<tr>
<td>WRITE_MEMORY</td>
<td>BOOL</td>
<td>• Activates the routine for writing the RFID data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>BOOL</td>
<td>• Activates the connection termination routine to the reader</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reacts to a positive edge</td>
</tr>
</tbody>
</table>
3 Functional Mechanisms of this Application

3.1 Program overview

Table 3-2 Description of the parameters of the RF_VAR structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_ADR (DWORD)</td>
<td>IP address of the RFID reader</td>
</tr>
<tr>
<td>DEVICE_ID (BYTE)</td>
<td>DEVICE_ID of the S7 CPU (see [6])</td>
</tr>
<tr>
<td>CON_ID (BYTE)</td>
<td>Unique connection ID for the TCP connection</td>
</tr>
<tr>
<td>LOCAL_PORT (WORD)</td>
<td>Local port</td>
</tr>
</tbody>
</table>
3 Functional Mechanisms of this Application

3.1 Program overview

Reading from and Writing to Transponders with RF670R

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAG_ID (STRING)</td>
<td>ID of the transponder to be read from/written to. (Hex format example of a 96-bit tag ID: 3005FB63AC1F3681EC880468)</td>
</tr>
<tr>
<td>new_TAG (STRING)</td>
<td>The new transponder ID with which the transponder is to be written to (Hex format example of a 96-bit tag ID: 3005FB63AC1F3681EC880468)</td>
</tr>
<tr>
<td>Source (STRING)</td>
<td>Name of the data source. This entry must match the entry in RF-MANAGER Basic.</td>
</tr>
<tr>
<td>Value_bank (INT)</td>
<td>Memory area of the transponder: 0 Reserved 1 EPC 2 TID 3 USER MEMORY Default value: 3.</td>
</tr>
<tr>
<td>value_startAddress (INT)</td>
<td>Number of the first byte in the memory area of the transponder</td>
</tr>
<tr>
<td>value_dataLength (INT)</td>
<td>Number of bytes to be read in. The maximum length of the data to be read per job is 64 bytes.</td>
</tr>
<tr>
<td>WRITE_DATA (ARRAY [1..128] of CHAR)</td>
<td>Data to be written to the memory area of the transponder. Each nibble (half a byte) must be represented as a hexadecimal character (see Figure 3-4). Example: You want to write the bytes 0x12 and 0x34 to the memory area of the transponder. To do so, you have to store the bytes as characters in DB RF_PARAMETER as follows: RF_VAR.WRITE_DATA[1] = '1' RF_VAR.WRITE_DATA[2] = '2' RF_VAR.WRITE_DATA[3] = '3' RF_VAR.WRITE_DATA[4] = '4' In this example, value_dataLength has the value 2.</td>
</tr>
<tr>
<td>Duration (INT)</td>
<td>Read duration in ms. Indicates how long the reader must read the transponder IDs. The command will be active for the entire duration.</td>
</tr>
<tr>
<td>TAG_Ids (INT)</td>
<td>Number of transponder IDs in the field that were read by the reader.</td>
</tr>
<tr>
<td>LNG (INT)</td>
<td>Length of the received byte data (*1)</td>
</tr>
</tbody>
</table>
3 Functional Mechanisms of this Application

3.1 Program overview

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REC_DATA (ARRAY [1..150] of CHAR)</td>
<td>Received data. Example: RF_VAR.REC_DATA [1]:='1' RF_VAR.REC_DATA [2]:='2' RF_VAR.REC_DATA [3]:='3' RF_VAR.REC_DATA [4]:='4'. The LNG of the RFID data has the value 2 and the received data has the values 0x12, 0x34 (*2).</td>
</tr>
</tbody>
</table>

(*1): If you receive more than one transponder ID, the length of the last transponder ID will be copied to the LNG field.

(*2): Max. up to 5 transponder IDs can be read for this field length. If you want to read more transponder IDs, you have to adjust this length (see Chapter 6.1).

The following figure shows how each nibble (half a byte) is to be represented as a hexadecimal character.

Figure 3-4 Coding specification for RF670R transponder

User format

Coding for RF670R transponder

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 63</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st nibble</td>
<td>1</td>
<td>1st nibble</td>
</tr>
<tr>
<td>2nd nibble</td>
<td>2</td>
<td>2nd nibble</td>
</tr>
<tr>
<td>3rd nibble</td>
<td>3</td>
<td>...............</td>
</tr>
<tr>
<td>4th nibble</td>
<td>4</td>
<td>1st nibble</td>
</tr>
</tbody>
</table>

Note

For a more detailed overview of the RFID-relevant parameters, please refer to the RF670R Function Manual (/5/, Chapter 3).
3.1.3 Call example: FB RF670R_RW (FB670) in FB CALL_RF670R (FB1)

Function block FB RF670R_RW (FB670) is called in FB CALL_RF670R (FB1). FB CALL_RF670R is called cyclically in OB1. The figure below shows the call of FB RF670R_RW in FB CALL_RF670R (FB1).

Figure 3-5 Call of FB RF670R_RW (FB670)
3 Functional Mechanisms of this Application

3.1 Program overview

3.1.4 State diagram of FB RF670R_RW (FB670)

The following figures show the state diagrams of the above-described functions. The figures are read from left to right. The S7 station sends XML commands to the RF670R reader. The RF670R reader responds with an XML message that contains the required data. This message is evaluated and stored to the respective fields of DB RF_PARAMETER.

**Connect**

Figure 3-6

**Read transponder IDs**

Figure 3-7
3 Functional Mechanisms of this Application

3.1 Program overview

Write transponder ID
Figure 3-8

Read RFID data
Figure 3-9

Write RFID data
Figure 3-10
3 Functional Mechanisms of this Application

3.1 Program overview

Disconnect

Figure 3-11

The following chart shows the graphical representation of the time sequences in the FB RF670R_RW function block (FB670).

Figure 3-12
3.2 Explanation of the functions implemented in FB RF670R_RW

In the following chapters, we show you the details of the functions implemented in FB RF670R_RW (FB670). We explain the most important SCL code fragments. For a complete overview, the well-documented SCL code is available to you in the project.

3.2.1 “Connect or disconnect”

Overview of functions

Via the CONNECT input parameter, the FB RF670R_RW function block establishes the TCP/IP connection to the RF670R reader and sends the hostGreetings message.

Via the DISCONNECT input parameter, the FB RF670R_RW function block sends the hostGoodbye message and terminates the TCP/IP connection to the RF670 reader.

Sequence of the function

The diagrammatic representation below shows the steps FB RF670R uses to establish/terminate a connection to the reader.

Figure 3-13

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONNECT is triggered.</td>
</tr>
</tbody>
</table>
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The variables that are necessary to establish the TCP connection are read out of DB RF_PARAMETER. The TCON block establishes the connection.</td>
</tr>
<tr>
<td>3</td>
<td>The XML-formatted “hostGreetings” command is sent to the reader with TSEND. If FB RF670R_RW receives positive feedback from the reader via TRCV, connection establishment has been successfully completed.</td>
</tr>
<tr>
<td>4</td>
<td>DISCONNECT is triggered.</td>
</tr>
<tr>
<td>5</td>
<td>The XML-formatted “hostGoodbye” command is sent to the reader with TSEND. If FB RF670R_RW receives positive feedback from the reader via TRCV, TDISCON will be called.</td>
</tr>
<tr>
<td>6</td>
<td>The TDISCON block terminates the TCP connection.</td>
</tr>
</tbody>
</table>

Program details

In this section, we show you the most important code fragments of this function from the documented source code of this example.

1. Triggering the “CONNECT” input

In the SCL code, a positive edge is generated to check the signal changes of the “CONNECT” input. When a rising edge of the input is detected, the block will start establishing the connection.

```
IF CONNECT AND NOT Connect_Edge THEN
  ... // Code fragments here...
```

2. Establishing the TCP connection

To parameterize the communication connections for TCP, the data structure from UDT 65 “TCONPAR” is created in the instance DB of FB RF670R_RW. This data structure contains the parameters you need to establish the connection. The TCON block is called and the TCP connection is established.

```
FB_CONNECT_REQ := REQ_CONN
  ID := RF_PARAM_CONN_ID
  CONNECT := TCONPAR);
```
3. Formatting the “hostGreetings” XML_command and sending it to the reader with TSEND

The “hostGreetings” command must have the following structure:

Figure 3-16

```
<frame>
  <cmd>
    <id> value_id </id>
    <hostGreetings>
      <readerType> value_readerType </readerType> opt
      <readerMode> value_ReaderMode </readerMode> opt
      <supportedVersions>
        <version> value_version </version> opt
        <version> value_version </version> opt
      </supportedVersions>
    </hostGreetings>
  </cmd>
</frame>
```

*opt* Optional: line can be omitted.

The figure below shows how the “hostGreetings” structure was implemented in the SCL code.

Figure 3-17

**Encoding of the individual constant XML substrings**

- `MESSAGE:=CONCAT(IN1 := START_XML ,IN2 := Val_ID);`
- `MESSAGE:=CONCAT(IN1 := MESSAGE ,IN2 := GREETING_STR1);`
- `MESSAGE:=CONCAT(IN1 := MESSAGE ,IN2 := GREETING_STR2);`
- `MESSAGE:=CONCAT(IN1 := MESSAGE ,IN2 := GREETING_STR3);`

**Copying the hostGreeting string to the send mailbox**

- `BLKMOVE:=BLKMOV(SRCBLK := MESSAGE,DSTBLK := MESSAGE_A);`

**Triggering the TSEND block**

- `FB_SEND(REQ :=REQ_SEND ,ID :=RF_PARAM.CON_ID ,LEN :=LENGTH ,DATA :=MESSAGE_A);`
3.2 Explanation of the functions implemented in FB RF670R_RW

4. Triggering the “DISCONNECT” input

In the SCL code, a positive edge is generated to check the signal changes of the “DISCONNECT” input. When a rising edge of the input is detected, the block will start terminating the connection to the reader.

Figure 3-18

```
IF DISCONNECT Edge NOT Disconnect_edge THEN
  REQ_SEND:=false;
  REQ_RCV:=true;
  REQ_DIS:=false;
  IF last_state_IDLE THEN
    DONE:=false;
    STATUS:=false;
    ERROR:=false;
    IF TCP_Control=true THEN
      busy:=true;
      n_state:=XML STR;  // switch to state XML STR
      trg_state:=DISCONNECT;
    ELSE
      temp_status_1:=W16#0000;  // identifier for RF670R_RW block
      temp_status_2:=W16#0100;  // no valid TCP Connection
      n_state:=ERROR_STATE;  // switch to ERROR state
    END_IF;
  ELSE
    temp_status_1:=W16#0100;  // identifier for RF670R_RW block
    temp_status_2:=W16#0102;  // FB is still active
    n_state:=ERROR_STATE;  // switch to ERROR state
  END_IF;
END_IF;
```

5. Formatting the “hostGoodbye” XML_command and sending it to the reader with TSEND

The “hostGoodbye” command must have the following structure:

Figure 3-19

```
<frame>
  <cmd>
    <id value_id /></id>
    <hostGoodbye>
      <readerMode value_ReaderMode /></readerMode>
    </hostGoodbye>
  </cmd>
</frame>
```

The figure below shows how the “hostGoodbye” structure is implemented in the SCL code.
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

Reading from and Writing to Transponders with RF670R

Figure 3-20

Encoding of the individual constant XML substrings

\[
\text{MESSAGE := CONCAT(IN1 := START_XML, IN2 := Val_ID);} \\
\text{MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := Goodbye_STR);} \\
\]

Copying the hostGoodbye string to the send mailbox

\[
\text{BLKMOVE := BLKMOV(SRCBLK := MESSAGE, DSTBLK := MESSAGE_A);} \\
\]

Triggering the TSEND block

\[
\text{FB SEND (REQ := REQ_SEND, ID := FF_PARAM.CON_ID, LEN := LENGTH, DATA := MESSAGE_A);} \\
\]

6. Terminating the TCP connection

If FB RF670R_RW receives positive feedback from the reader via TRCV, TDISCON will be called and the connection will be terminated.

Figure 3-21

\[
\text{FB_DISCONNECT (REQ := REQ_DIS, ID := RF_PARAM.CON_ID);} \\
\text{IF FB_DISCONNECT.DONE THEN} \\
\text{IF last_state = Time_OUT THEN} \\
\text{DONE := false; n_state := IDLE;} \\
\text{elsIF last_state = No_CON THEN} \\
\text{n_state := CONNECTION; last_state := IDLE; Value_id := 0;} \\
\text{ELSE} \\
\text{TCP_connect := false; REQ_DIS := false; BUSY := FALSE; DONE := true;} \\
\text{n_state := IDLE; END IF;} \\
\]

3.2.2 “Read or write transponder IDs”

Overview of functions

The FB RF670R_RW function block generates a readTagIDs command and sends it to the RF670R reader.

All transponder IDs of the transponders the RF670R could read are now sent to the CPU. They are filtered and stored in the REC_DATA area of DB RF_PARAMETER.

Via the WRITE_TAG_ID input parameter, the FB RF670R_RW function block generates a writeTagID command and sends it to the RF670R reader to write the transponder ID of the transponder in the field. If a positive response message is received, the WRITE process is successfully completed.
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

**Sequence of the function**

The diagrammatic representation below shows the steps FB RF670R uses to read/write the transponder ID.

Figure 3-22

**Read/write transponder IDs**

Table 3-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>READ_TAG_ID or WRITE_TAG_ID is triggered.</td>
</tr>
<tr>
<td>2</td>
<td>The “readTagID” or “writeTagID” XML command is formatted.</td>
</tr>
<tr>
<td>3</td>
<td>The XML-formatted command is sent to the reader in TSEND.</td>
</tr>
<tr>
<td>4</td>
<td>If FB RF670R_RW receives positive feedback from the reader via TRCV, the received data will be evaluated and copied to DB RF_PARAMETER.</td>
</tr>
</tbody>
</table>

**Program details**

In this section, we show you the most important code fragments of this function from the documented source code of this example.

1. **Triggering the “READ_TAG_ID” or “WRITE_TAG_ID” input**

In the SCL code, a positive edge is generated to check the signal changes of the “READ_TAG_ID” or “WRITE_TAG_ID” input. When a rising edge of the input is detected, the block will start reading or writing the transponder_ID.
3.2 Explanation of the functions implemented in FB RF670R_RW

2. Formatting the “readTagID” or “writeTagID” XML command

The readTagIDs command has the following structure:

Figure 3-24

```xml
<frame>
  <cmd>
    <id> value_id </id>
    <readTagIDs>
      <sourceName> value_sourceName </sourceName>
      <duration> value_duration </duration>
    </readTagIDs>
  </cmd>
</frame>
```

The figure below shows how the “readTagID” structure was implemented in the SCL code.
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

Figure 3-25

**Encoding of the individual constant XML substrings**

```
MESSAGE:=CONCAT(IN1 := START_XML,
                  IN2 := Val_ID);
MESSAGE:=CONCAT(IN1 := MESSAGE,
                  IN2 := Read_TAG_ID1);
MESSAGE:=CONCAT(IN1 := MESSAGE,
                  IN2 := source);
MESSAGE:=CONCAT(IN1 := MESSAGE,
                  IN2 := Read_TAG_ID2);
MESSAGE:=CONCAT(IN1 := MESSAGE,
                  IN2 := Duration);
MESSAGE:=CONCAT(IN1 := MESSAGE,
                  IN2 := Read_TAG_ID3);
```

**Copying the readTagID string to the send mailbox**

```
BLKMOVE:=BLKMOV(SRCBLK := MESSAGE, DSTBLK := MESSAGE_i);
```
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

The writeTagIDs command has the following structure:

```
<frame>
<cmd>
  <id> value_id </id>
  <writeTagID>
    <sourceName> value_sourceName </sourceName>
    <tagID> value_tagID </tagID> opt
    <newID> value_newID </newID> opt
    <idLength> value_idLength </idLength> opt
    <password> value_password </password> opt
  </writeTagID>
</cmd>
</frame>
```

Optional: line can be omitted.

The figure below shows how the “writeTagID” structure was implemented in the SCL code.

```
3. Sending the XML-formatted command to the reader

FB_SEND (REQ := REQ_SEND
  ,ID := RF_PARAM.CON_ID
  ,LEN := LENGTH
  ,DATA := MESSAGE_A);
```

4. Receiving and evaluating data

When the response message is received, the read or written transponder IDs will be extracted and stored in the global data block RF_PARAMETER.
The response message for the ReadTagID command has the following structure:
Figure 3-29

```c
<frame>
  <id> value_id </id>
  <resultCode> 0 </resultCode>
  <readTagIDs>
    <returnValue>
      <tag>
        <tagID> value_tagID </tagID>
        <utcTime> value_utcTime </utcTime> opt
        <antennaName> value_antennaName </antennaName> opt
        <rSSI> value_rSSI </rSSI> opt
      </tag>
      ...
      <tag> opt
      </tag> opt
      </tag>
      </tag>
    </returnValue>
  </readTagIDs>
</reply>
</frame>
```

*Optional: line could be omitted.*

The following code fragment shows how the individual transponder IDs are extracted from the receive buffer and written to the respective RF_PARAMETER area.
Figure 3-30

```c
// search complete receiving buffer for partial pattern "tag=tagID"
// this pattern can occur more than once in the receiving buffer IF more
// than one transponder is in the RF/UR field
Receive_Point:=RF_PARAM.REC_DATA[NG:ID];
offset:=Rec.dataGrpTable;
FOR I:=1 TO J+1 DO
  IF rec_data[1]="g" AND rec_data[1]="g" AND rec_data[1]="g" AND rec_data[1]="g" THEN
    n_Tag:=1+1; // remember location in rec_data
    ID_TAGNR:=TAG_NR+1;
    FOR J:=n_tag TO n_tag+8 DO
      IF rec_data[2]="g" AND rec_data[2]="g" THEN
        LNG_ID:=J-n_tag;
      EXIT;
    END_IF;
  END_FOR;
  Receive_Point:=RF_PARAM.REC_DATA;
  LSN_REC:=Rec.Length;
  Rec.dataPointer:=offset;
  Any_Point:=rec_data[n.Tag];
  variable.length:=LNG_ID;
  rec_length:=LNG_ID;
  RF_PARAM.LNG:=LNG_ID;
  TAG_LEN:=LNG_ID;
  LNG_ID:=TAG_NR+LNG_ID+TAG_NR; // new offset off received Data buffer
  BLKMOV(BLKMOV(BLKMOV(Rec.LNG := Any_Point, LSTELK := Receive_Point));
  RF_PARAM.TAG_ID:=TAG_NR;
  EXIT;
END_IF;
```

Copyright © Siemens AG 2012 All rights reserved
3.2 Explanation of the functions implemented in FB RF670R_RW

3.2.2 Reading from and Writing to Transponders with RF670R

The response message for the WriteTagID command has the following structure:

Figure 3-32

```xml
<frame>
  <reply>
    <id>value id</id>
    <resultCode>0</resultCode>
    <setTagID/>
  </reply>
</frame>
```

If FB RF670R_RW receives positive feedback from the reader via TRCV, the new_TAGid will be copied to DB RF_PARAMETER.

Figure 3-33

```c
LNG_ID:=LEN|S:= New ID|
BF_PARAM LANG:= LNG_Tid
ELKMOV:=ELKMOV(SRCBLK := RF_PARAM.new_ID,DSTBLK := RF_PARAM.ESC_DATA );
```

3.2.3 “Read RFID data from or write RFID data to transponder”

Overview of functions

With the aid of these functions, payload can be read from or written to the transponder memory.

The readTagMemory message is used to read from and the writeTagMemory message is used to write to the transponder.
When there are several transponders in the read field, the following variants apply.

**Variant 1:** TAG_ID field in DB RF_PARAMETER is blank:
- The payload of the transponder the reader detects first is read or written.

**Variant 2:** TAG_ID field in DB RF_PARAMETER exists:
- Only data of the transponder with the specified transponder ID is read/written.

**Sequence of the function**

![Sequence diagram](image-url)

**Table 3-5**

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>READ_MEMORY or WRITE_MEMORY is triggered.</td>
</tr>
<tr>
<td>2</td>
<td>The readTagMemory or writeTagMemory command is formatted.</td>
</tr>
<tr>
<td>3</td>
<td>The XML-formatted command is sent to the reader in TSEND.</td>
</tr>
<tr>
<td>4</td>
<td>If FB RF670R_RW receives positive feedback from the reader via TRCV, the received data will be evaluated and copied to DB RF_PARAMETER.</td>
</tr>
</tbody>
</table>

**Program details**

In this section, we show you the most important code fragments of this function from the documented SCL code of this example.

1. **Triggering the “READ_MEMORY” or “WRITE_MEMORY” input**

   In the SCL code, a positive edge is generated to check the signal changes of the “READ_MEMORY” or “WRITE_MEMORY” input. When a rising edge of the input is detected, the block will start reading or writing the RFID data.
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

Figure 3-35

```plaintext
IF READ_MEMORY AND NOT _N_Mem_Edge THEN
  IF test_state=FULL THEN
    IF TCP_Connect=true THEN
      BUSY:=TRUE;
      n_state:=XML_STR;  // switch to state XML_STR
      ctry_state:=N_MEMORY;
    ELSE
      temp_status_1:=#16#0000;
      temp_status_1:=#16#0104;  // no valid TCP Connection
      n_state:=ERROR_STATE;  // switch to ERROR state
    END_IF;
    ELSE
      temp_status_1:=#16#0000;
      temp_status_2:=#16#8102;  // FB is still active
      n_state:=ERROR_STATE;  // switch to ERROR state
    END_IF;
  END_IF;
  _N_Mem_Edge:=READ_MEMORY;  // Edge Detector

OR

IF WRITE_MEMORY AND NOT _N_Mem_Edge THEN
  IF test_state=FULL THEN
    IF TCP_Connect=true THEN
      BUSY:=TRUE;
      n_state:=XML_STR;  // switch to state XML_STR
      ctry_state:=N_MEMORY;
    ELSE
      temp_status_1:=#16#0000;
      temp_status_2:=#16#8104;  // no valid TCP Connection
      n_state:=ERROR_STATE;  // switch to ERROR state
    END_IF;
    ELSE
      temp_status_1:=#16#0000;
      temp_status_2:=#16#8102;  // FB is still active
      n_state:=ERROR_STATE;  // switch to ERROR state
    END_IF;
  END_IF;
  _N_Mem_Edge:=WRITE_MEMORY;  // Edge Detector
```

Reading from and Writing to Transponders with RF670R
V1.0, Entry ID: 23626344

37
2. Formatting the “readTagMemory” or “writeTagMemory” XML command

The readTagMemory command has the following structure:

Figure 3-36

```
<frame>
  <cmd>
    <id> value_id </id>
    <readTagMemory>
      <sourceName> value_sourceName </sourceName>
      <tagID> value_tagID </tagID> opt
      <password> value_password </password> opt
      <tagField>
        <bank> value_bank </bank>
        <startAddress> value_startAddress </startAddress>
        <dataLength> value_dataLength </dataLength>
      </tagField>...
    </readTagMemory>
  </cmd>
</frame>
```

*opt* Optional: line can be omitted.

The figure below shows how the “readTagMemory” structure was implemented in the SCL code.

Figure 3-37

Encoding of the individual constant XML substrings

```
MESSAGE := CONCAT(IN1 := START.XML, IN2 := Val.ID);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY1);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := source);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY2);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := TAG_ID);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY3);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := Bank);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY4);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := Value_start_kdr);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY5);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := VAL_DAT_LNG);
MESSAGE := CONCAT(IN1 := MESSAGE, IN2 := READ_MEMORY6);
```

Copying the readTagMemory string to the send mailbox

```
BLKMVC := ELKMVC(SRCBLK := MESSAGE, DSTBLK := MESSAGE_A); // Copy Stringbuffer
```
3.2 Explanation of the functions implemented in FB RF670R_RW

The writeTagMemory command has the following structure:

```xml
<frame>
  <cmd>
    <id> value_id </id>
    <writeTagMemory>
      <sourceName> value_sourceName </sourceName>
      <tagID> value_tagID </tagID> opt
      <password> value_password </password> opt
      <tagField>
        <bank> value_bank </bank>
        <startAddress> value_startAddress </startAddress>
        <dataLength> value_dataLength </dataLength>
        <data> value_data </data> opt
      </tagField>
    </writeTagMemory>
  </cmd>
</frame>
```

(Optional: line can be omitted)

The figure below shows how the “writeTagMemory” structure was implemented in the SCL code.

Encoding of the individual constant XML substrings

```plaintext
MESSAGE := CONCAT(IN1 := START_XML
                   , IN2 := Val_ID);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := WRITE_MEMORY1);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := source);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := READ_MEMORY2);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := TAG_ID);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := READ_MEMORY3);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := Bank);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := READ_MEMORY4);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := Value_start_Adr);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := READ_MEMORY5);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := VAL_DAT_LEN);
MESSAGE := CONCAT(IN1 := MESSAGE
                   , IN2 := WRITE_MEMORY2);
LENGTH := LEN(MESSAGE);
BLKMOVE := BLKMOV(SRCBLK := MESSAGE, DSTBLK := MESSAGE_A); // Copy St:
Any_Point := MESSAGE_A[LENGTH=1];
variable.length := RF_PAPAM.value_dataLength%;
BLKMOVE := BLKMOV(SRCBLK := RF_PAPAM.WRITE_DATA, DSTBLK := Any_Point);
Any_Point := MESSAGE_A[LENGTH=variable.length];
Copying the writeTagMemory string to the send mailbox
MESSAGE := WRITE_MEMORY3;
LENGTH := LEN(MESSAGE);
variable.length := LENGTH;
BLKMOVE := BLKMOV(SRCBLK := MESSAGE, DSTBLK := Any_Point); // copy str:
```

Reading from and Writing to Transponders with RF670R

V1.0, Entry ID: 23626344
3 Functional Mechanisms of this Application

3.2 Explanation of the functions implemented in FB RF670R_RW

3. Sending the XML-formatted command to the reader

Figure 3-40

```
FB_SEND(REQ :=REQ_SEND,
        ID :=RF_PARAM.CON_ID,
        LEN :=LENGTH,
        DATA:=MESSAGE_A);
```

4. Receiving and evaluating data

When the response message is received, the read data will be extracted and stored in the global data block RF_PARAMETER.

The response message for the ReadTagMemory command has the following structure:

Figure 3-41

```xml
<frame>
  <reply>
    <id> value_id </id>
    <resultCode> 0 </resultCode>
    <readTagMemory>
      <returnValue>
        <tag>
          <tagID> value_tagID </tagID>
          <success> value_success </success>
          <utcTime> value_utcTime </utcTime> opt
          <antennaName> value_antennaName </antennaName> opt
          <rSSI> value_rSSI </rSSI> opt
          <tagField> opt
            <bank> value_bank </bank>
            <startAddress> value_startAddress </startAddress>
            <dataLength> value_dataLength </dataLength>
            <data> value_data </data>
          </tagField> opt
          ...<tagField> opt
          ...
        </tag>
        ...<tag> opt
        ...
      </returnValue>
    </readTagMemory>
  </reply>
</frame>
```

opt: Optional line could be omitted.
The following code fragment shows how the data is extracted from the receive buffer and written to the respective RF_PARAMETER area.

Figure 3-42

```c
// searching first 500 characters of receiving buffer rec data for pattern "<data"
FOR i := 0 TO 500 DO
  IF rec_data[i]="<" AND rec_data[i+1]="=" AND rec_data[i+2]="=" AND rec_data[i+3]="=" THEN
    ANY Point := rec_data[i+4];
    offset := variable.data_pointer; // remember location
    ID := TRUE; // pattern found
    EXIT;// EXIT Loop
  ELSE // pattern not found
    temp_status := #8160000; // RFID ERROR
    temp_status := #8160001; // TAG NOT FOUND
    n_state := ERROR_STATE; // switch to ERROR state
  END IF;
END FOR;

IF ID TRUE THEN // "<data>"-Pattern found in receiving buffer
  // extract the meta data from receive buffer and copy to RF_PARAM
  Any point := rec_data;
  variable.data_pointer := offset;
  variable.length := length ID;
  BLKMOV := ELKMOV(SRCBLK := Any point, DSTBLK := RF PARAM.REC DATA);

Figure 3-43 RF_PARAMETER
```

<table>
<thead>
<tr>
<th>RF_VAR.value_startAddress</th>
<th>0000000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_VAR.value_DataLength</td>
<td>0000000000</td>
</tr>
<tr>
<td>RF_VAR.Duraction</td>
<td>0000000000</td>
</tr>
<tr>
<td>RF_VAR.UNITX_DATA(1)</td>
<td>0000000000</td>
</tr>
<tr>
<td>RF_VAR.UNITX_DATA(2)</td>
<td>0000000000</td>
</tr>
<tr>
<td>RF_VAR.UNITX_DATA(3)</td>
<td>0000000000</td>
</tr>
<tr>
<td>RF_VAR.UNITX_DATA(4)</td>
<td>0000000000</td>
</tr>
</tbody>
</table>

Length of the data
RFID data to be written to the transponder

In HEX format, the payload corresponds to 0x12 and 0x9F. The length of the data has the value 2.

The response message for the WriteTagMemory command has the following structure:

Figure 3-44
3 Functional Mechanisms of this Application

3.3 Error and status display

The following code fragment shows how the data from the receive buffer is evaluated.

Figure 3-45

```c
// searching first 500 characters of receiving buffer rec_data for pattern "success<7c"
FOR i := 0 TO 500 DO
   // detect the pattern
      ID True:= TRUE; // pattern found
   END IF;
END FOR;
```

Note: For more program details, the well-documented SCL code is available to you in the project (see \( \))

3.3 Error and status display

For error diagnostics, the FB RF670R_RW function block (FB670) has a STATUS output. By reading the STATUS output of the function block, you are provided with information on logical errors and error messages that may occur during the communication between the controller and the RF670R reader.

3.3.1 Error messages of FB RF670R_RW (FB670)

Table 3-6

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
<th>Support/remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#00008101</td>
<td>Watchdog timer has expired.</td>
<td>1. a) Check IP_ADR input parameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b) Check communication between controller and reader</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Re-trigger CONNECT</td>
</tr>
<tr>
<td>16#00008102</td>
<td>The previous job has not yet been completed.</td>
<td>1. Wait until BUSY=FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Restart process</td>
</tr>
<tr>
<td>16#00008103</td>
<td>There is already a connection to the reader.</td>
<td>1. Check transponder ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Restart process</td>
</tr>
<tr>
<td>16#00008104</td>
<td>There is no connection to the reader.</td>
<td>Trigger CONNECT</td>
</tr>
<tr>
<td>16#00008105</td>
<td>No transponder in the field or incorrect transponder ID.</td>
<td>1. Check transponder ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Restart process</td>
</tr>
<tr>
<td>16#00008106</td>
<td>Incorrect response message (received Value_id does not equal sent Value_id).</td>
<td>1. Trigger DISCONNECT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Trigger CONNECT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Restart process</td>
</tr>
<tr>
<td>16#00008107</td>
<td>The WRITE_TAG_ID process could not be successfully completed.</td>
<td>Restart process</td>
</tr>
<tr>
<td>16#00008108</td>
<td>The length of the evaluated characters exceeds 150 characters. Only 150 characters were transmitted.</td>
<td>1. Change length of RF_VAR.REC_DATA in DB RF_PARAMETER (see Chapter 6.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Restart process</td>
</tr>
</tbody>
</table>
3.3 Error and status display

### 3.3.2 Error messages of the RF670R reader

Errors with the **16#13000xyy** status are RFID errors. For a more detailed overview of the result codes, please refer to the RF670R Function Manual (/5/ Chapter 3.3).

### 3.3.3 Error messages of communication blocks

Errors with a status that is not described above are errors of communication blocks (see /8/):

- **TCON**:  \( \text{DW#16#0001xyyy} \)
- **TSEND**:  \( \text{DW#16#0010xyyy} \)
- **TRCV**:  \( \text{DW#16#0012xyyy} \)
- **TDISCON**:  \( \text{DW#16#0011xyyy} \)

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
<th>Support/remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>16#00008109</td>
<td>The connection is interrupted, for example due a line break</td>
<td>1. Check communication between controller and reader 2. Re-trigger CONNECT</td>
</tr>
</tbody>
</table>
4 Installation and Commissioning

4.1 Hardware configuration

For the necessary hardware components, please refer to chapter 2.3.

NOTICE
Follow the installation guidelines for S7-300 (17) and RF670R (14). Refer to the relevant manuals.

NOTICE
Before you switch on the power supply, complete and check the installation!

The figure below shows the hardware configuration of the application.
Figure 4-1

The following table provides an overview of the IP addresses used in this sample program.
Table 4-1

<table>
<thead>
<tr>
<th>Module</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 315-2PN/ DP</td>
<td>192.168.0.1</td>
</tr>
<tr>
<td>PC/ PG</td>
<td>192.168.0.3</td>
</tr>
<tr>
<td>RF670R</td>
<td>192.168.0.254</td>
</tr>
</tbody>
</table>
4.2 Hardware installation: S7 station

Table 4-2

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Attach the individual modules to a suitable rack.</td>
<td>Table 2-1: Hardware components</td>
</tr>
<tr>
<td>2</td>
<td>Connect the PS307 to the network. (230 V AC)</td>
<td>Ensure that the polarity is correct.</td>
</tr>
<tr>
<td>3</td>
<td>Connect the following devices:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PROFINET interface of the engineering PG to the PROFINET interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of the CPU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PROFINET interface of the RFID reader to the second PROFINET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interface of the CPU</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Hardware installation: RF670R reader

Table 4-3

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connect the antennas to the respective sockets.</td>
<td>You have to configure the antennas using RF-MANAGER.</td>
</tr>
<tr>
<td>2</td>
<td>Connect the wide-range power supply unit to the network.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Connect the RF670R reader to the wide-range power supply unit.</td>
<td></td>
</tr>
</tbody>
</table>

4.4 Installation of the standard software

The engineering station is used as the configuration computer for the S7 station.

Table 4-4

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install STEP 7 V5.5.</td>
<td>Follow the instructions of the installation program.</td>
</tr>
<tr>
<td>2</td>
<td>Install S7-SCL V5.3+SP5.</td>
<td>Follow the instructions of the installation program.</td>
</tr>
<tr>
<td>3</td>
<td>Install SIMATIC RF-MANAGER Basic 2010.</td>
<td>Follow the instructions of the installation program.</td>
</tr>
</tbody>
</table>
4.5 Installing the STEP 7 project

The following table lists the steps necessary to install the sample code.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The project is available on the HTML page from which you downloaded this document. Save the “23626344_RF670R_CODE_V10.zip” project to your hard drive.</td>
</tr>
<tr>
<td>2</td>
<td>Open the SIMATIC MANAGER and retrieve the STEP 7 project. “File &gt; Retrieve...”</td>
</tr>
<tr>
<td>3</td>
<td>The project is now available to you.</td>
</tr>
</tbody>
</table>

4.6 Setting the PG/PC interface

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the SIMATIC MANAGER, set the PC interface to TCP/IP  “Options &gt; Set PC/PG Interface...”</td>
</tr>
<tr>
<td>2</td>
<td>Select the access path. For the used network card, select TCP/IP. Confirm with &quot;OK&quot;.</td>
</tr>
</tbody>
</table>

---

Copyright © Siemens AG 2012 All rights reserved
4.7 Configuring the PG/PC

Changing the IP address

The figure shows the network setting to which you have to change the PG/PC!

Table 4-7

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark/note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the Internet Protocol (TCP/IP) Properties by selecting “Start &gt; Settings &gt; Network Connection &gt;Local Connections”.&lt;br&gt;In the open window, select Internet Protocol (TCP/IP) and open the Properties.&lt;br&gt;Select “Use the following IP address” and fill out the field as shown in the screenshot. Close the dialog boxes with “OK”.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>If your PG has an IWLAN interface, disable it.</td>
<td></td>
</tr>
</tbody>
</table>
4.8 Configuring the S7 station

Changing the IP address of the CPU

Before the STEP 7 project can be downloaded to the CPU, you have to change the IP address of the S7-300/400 CPU as shown in Table 4-1 via which the project is downloaded to the CPU.

Table 4-8

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the SIMATIC Manager, open a STEP 7 project.</td>
</tr>
<tr>
<td>2</td>
<td>In the “PLC” menu, select the “Edit Ethernet Node…” option.</td>
</tr>
</tbody>
</table>

![Image of SIMATIC Manager with Ethernet Node option]
3. Click on the “Browse…” button.

4. Select the desired module and click on “OK” to confirm the selection.
### 4.8 Configuring the S7 station

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>In the “Set IP configurations” window that appears, enter the IP address as shown in Table 4-1. Click on the “Assign IP Configuration” button. Close the dialog box with the “Close” button.</td>
</tr>
</tbody>
</table>

![Image of the configuration window with IP address fields highlighted.]
### 4.9 Configuring the RF670R reader with RF-MANAGER Basic 2010

**Table 4-9**

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open RF-MANAGER Basic 2010 and create a new project.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Change the IP address of the RF670R “RFID device&gt;General&gt;IP address”.</td>
<td>This IP address must be entered in DB RF_PARAMETER “IP_ADR” (this has already been done in the STEP 7 project).</td>
</tr>
<tr>
<td>3</td>
<td>Enable only the first antenna “Antennas&gt;Antenna0X&gt;Enable”.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Set the power level “Antennas&gt;Antenna01&gt;Power level”.</td>
<td>The power level was increased to 500 mW. This is especially useful for writing to transponders.</td>
</tr>
</tbody>
</table>
## 4 Installation and Commissioning

### 4.9 Configuring the RF670R reader with RF-MANAGER Basic 2010

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Enter the name of the data source for the antenna “Sources&gt;Name”.</td>
<td>This name must match the entry in DB RF_PARAMETER “Source” (this has already been done in the STEP 7 project).</td>
</tr>
</tbody>
</table>

![Configuration screenshot](image1.png)

**Source_1 (Source)**

- **Name**: Source_1
- **Session**: 50

6. Download the configuration to the RF670R reader.

![Download configuration](image2.png)
4.10 Downloading the STEP 7 project

Prerequisite:
- General CPU reset has been performed (see Chapter 8.4.3).
- The SIMATIC Micro Memory Card (MMC) of the CPU has been deleted.

Table 4-10

<table>
<thead>
<tr>
<th>No.</th>
<th>Action</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the SIMATIC Manager, open the following STEP 7 project: &quot;RF670R_READ_WRITE&quot;.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Select the &quot;RF670R_PNCPU&quot; S7 station and download the entire project to your CPU.</td>
<td>&quot;PLC &gt; Download&quot;</td>
</tr>
</tbody>
</table>

**Note**
If you recompile all blocks, set the recompiled IDB_RF670R_RW instance data block to “Non Retain” to overwrite the respective instance DB with the initial values when restarting the CPU.

4.11 Orientation of the transponders with respect to the antenna

**Polarization axis**

As the RF620A antenna is linearly polarized, the correct orientation of the transponders used must be ensured with respect to the antenna. Generally, the polarization axes of antenna and transponders must run parallel to one another. The symbol on the antenna provides a graphical representation of how the polarization axis runs.

Figure 4-2 Polarization axis
Orientation

The figure below shows the optimum orientation of the RF640T transponders with respect to the RF620A antenna.

Figure 4-3 Antenna/transponder orientation
5 Operation of the Application

5.1 Overview

This chapter shows you how to operate the above-described functions of this application. All necessary variables can be found in the "VAT_RF_RW" variable table.

Figure 5-1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Connect</td>
<td>Connect</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Read_Tag_IDs</td>
<td>Read Transponder-IDs</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Write_Tag_id</td>
<td>Write Transponder-ID</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Read_Memory</td>
<td>Read RFID data</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Write_Memory</td>
<td>Write RFID data</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Disconnect</td>
<td>Disconnect</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.TCP_connect</td>
<td>Connection Status</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Done</td>
<td>Job has been completed without errors</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Error</td>
<td>Error</td>
</tr>
<tr>
<td>&quot;IDB_CALL_RF670R&quot;.Status</td>
<td>Error Status</td>
</tr>
</tbody>
</table>
5.2 Connecting to the RF670R

The table below lists instructions for establishing the connection to the RF670R reader.

Table 5-2 Connecting to the RF670R

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open DB RF_PARAMETER and check if all parameters necessary for establishing the connection have been entered. “View &gt; Data View &gt; Actual Value”</td>
</tr>
<tr>
<td></td>
<td>If changes are made, save and once again download the DB RF_PARAMETER data block.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Initial value</th>
<th>Actual value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 D</td>
<td>RF_VAR.IP_ADDR</td>
<td>DWORD</td>
<td>D#1660</td>
<td>D#1660CB0AFF</td>
</tr>
<tr>
<td>4.0 D</td>
<td>RF_VAR.DEVICE_ID</td>
<td>BYTE</td>
<td>D#16#2</td>
<td>D#16#2</td>
</tr>
<tr>
<td>6.0 W</td>
<td>RF_VAR.CON. ID</td>
<td>WORD</td>
<td>U#16#0</td>
<td>U#16#0</td>
</tr>
<tr>
<td>8.0 W</td>
<td>RF_VAR.LOCAL.POINT</td>
<td>WORD</td>
<td>U#16#0</td>
<td>U#16#0</td>
</tr>
<tr>
<td>10.0 W</td>
<td>RF_VAR.TAG.ID</td>
<td>STRING</td>
<td>{ ' ' ' ' }</td>
<td>{ ' ' ' ' }</td>
</tr>
</tbody>
</table>

2 | Open the "VAT_RF_RW" variable table and enable "IDB_CALL_RF670R". Connect to establish the connection to the RF670R reader. |

3 | If the connection was successfully established, "IDB_RF670R_RW".TCP_connect and "IDB_CALL_RF670R".Done will be set. |
## 5.3 Reading the transponder ID

The table below lists instructions for reading the transponder IDs.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the &quot;TRUE&quot; status (*1).</td>
</tr>
<tr>
<td></td>
<td><strong>DB3 DBX 0.4</strong> &quot;IDB_CALL_RF670R&quot;.Write_Memory <strong>BOOL</strong> false</td>
</tr>
<tr>
<td></td>
<td><strong>DB3 DBX 0.5</strong> &quot;IDB_CALL_RF670R&quot;.DisConnect <strong>BOOL</strong> true</td>
</tr>
<tr>
<td>2</td>
<td>Enable &quot;IDB_CALL_RF670R&quot;.Read_Tag_IDs to read the transponder IDs of the transponders in the field.</td>
</tr>
<tr>
<td></td>
<td><strong>DB670 DBX 3552.0</strong> &quot;IDB_RF670R_RW&quot;.TCP_connect <strong>BOOL</strong> true</td>
</tr>
<tr>
<td>3</td>
<td>If the process was successfully completed, &quot;IDB_CALL_RF670R&quot;.Done will be set.</td>
</tr>
<tr>
<td></td>
<td><strong>DB3 DBX 0.5</strong> &quot;IDB_CALL_RF670R&quot;.DisConnect <strong>BOOL</strong> false</td>
</tr>
<tr>
<td></td>
<td><strong>DB670 DBX 3352.0</strong> &quot;IDB_RF670R_RW&quot;.TCP_connect <strong>BOOL</strong> true</td>
</tr>
<tr>
<td></td>
<td><strong>DB3 DBX 0.6</strong> &quot;IDB_CALL_RF670R&quot;.Done <strong>BOOL</strong> true</td>
</tr>
<tr>
<td></td>
<td><strong>DB3 DBX 0.7</strong> &quot;IDB_CALL_RF670R&quot;.Error <strong>BOOL</strong> false</td>
</tr>
</tbody>
</table>
5 Operation of the Application

5.4 Writing the transponder ID

The table below lists instructions for writing the transponder ID.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the &quot;TRUE&quot; status (*1).</td>
</tr>
<tr>
<td>2</td>
<td>Enter the current and the new transponder ID in RF_PARAMETER to write the transponder ID of a transponder.</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.

---

Table 5-4 Writing the transponder ID

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Open DB RF_PARAMETER. If the read operation has been successfully completed, the transponder IDs must be stored in the RF_VAR.REC_DATA field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the &quot;TRUE&quot; status (*1).</td>
</tr>
<tr>
<td>2</td>
<td>Enter the current and the new transponder ID in RF_PARAMETER to write the transponder ID of a transponder.</td>
</tr>
</tbody>
</table>

Save and once again download the DB RF_PARAMETER data block.
5 Operation of the Application

5.4 Writing the transponder ID

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Enable &quot;IDB_CALL_RF670R&quot;.Write_Tag_ID.</td>
</tr>
<tr>
<td>4</td>
<td>If the process was successfully completed, &quot;IDB_CALL_RF670R&quot;.Done will be set.</td>
</tr>
<tr>
<td>5</td>
<td>Follow steps 1-2 of Table 5-2 to check if the new transponder ID was successfully written to the transponder. Result in DB RF_PARAMETER:</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.
## 5.5 Reading RFID data from a transponder

The table below lists instructions for reading RFID data from a transponder.

### Table 5-5 Reading RFID data from a transponder

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the “TRUE” status (*1).</td>
</tr>
<tr>
<td>2</td>
<td>Enter the tag ID of the transponder whose memory you want to read in RF_PARAMETER.</td>
</tr>
<tr>
<td>3</td>
<td>Enable &quot;IDB_CALL_RF670R&quot;.Read_Memory.</td>
</tr>
<tr>
<td>4</td>
<td>If the process was successfully completed, &quot;IDB_CALL_RF670R&quot;.Done will be set.</td>
</tr>
<tr>
<td>5</td>
<td>Open DB RF_PARAMETER. If the read operation has been successfully completed, the RFID data must be stored in the RF_VAR.REC_DATA field.</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.
5.6 Writing RFID data to a transponder

The table below lists instructions for writing RFID data to a transponder.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the &quot;TRUE&quot; status (*1).</td>
</tr>
<tr>
<td>2</td>
<td>Enter the transponder ID of the transponder whose memory you want to write, the length of the RFID data and the RFID data in RF_PARAMETER. Save and once again download the DB RF_PARAMETER data block.</td>
</tr>
<tr>
<td>3</td>
<td>Enable &quot;IDB_CALL_RF670R&quot;.Write_Memory.</td>
</tr>
<tr>
<td>4</td>
<td>If the process was successfully completed, &quot;IDB_CALL_RF670R&quot;.Done will be set.</td>
</tr>
</tbody>
</table>

*1: 
| DB3.DBX 0.4 | "IDB_CALL_RF670R".Write_Memory | BOOL | false |
| DB3.DBX 0.5 | "IDB_CALL_RF670R".Disconnect | BOOL | false |
| DB3.DBX 0.6 | "IDB_CALL_RF670R".Done | BOOL | false |

*2: 
| DB670.DBX 3352.0 | "IDB_RF670R_RW".TCP_connect | BOOL | true |
| DB3.DBX 0.7 | "IDB_CALL_RF670R".Error | BOOL | false |
5 Operation of the Application

5.7 Disconnecting

Follow steps 1-3 of Table 5-4 to check if the new RFID data was successfully written to the transponder.

Result in DB RF_PARAMETER:

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>456.0</td>
<td>RF_VAR.LNG</td>
</tr>
<tr>
<td>458.0</td>
<td>RF_VAR.REC_DATA[1]</td>
</tr>
<tr>
<td>459.0</td>
<td>RF_VAR.REC_DATA[2]</td>
</tr>
<tr>
<td>460.0</td>
<td>RF_VAR.REC_DATA[3]</td>
</tr>
<tr>
<td>461.0</td>
<td>RF_VAR.REC_DATA[4]</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.

### 5.7 Disconnecting

The table below provides instructions for terminating the connection.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check if &quot;IDB_RF670R_RW&quot;.TCP_connect has the &quot;TRUE&quot; status (*1).</td>
</tr>
<tr>
<td>5</td>
<td>DB3.DBDX 0.4 &quot;IDB_CALL_RF670R&quot;.Write_Memory BOOL false</td>
</tr>
<tr>
<td>6</td>
<td>DB3.DBDX 0.5 &quot;IDB_CALL_RF670R&quot;.Disconnect BOOL false</td>
</tr>
<tr>
<td>7</td>
<td>DB3.DBDX 6.0 &quot;IDB_CALL_RF670R&quot;.STATUS_Conv HEX 0000000000000001</td>
</tr>
<tr>
<td>8</td>
<td>DB670.DBX 3352.0 &quot;IDB_RF670R_RW&quot;.TCP_connect BOOL true</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.

2 | Enable "IDB_CALL_RF670R".Disconnect to disconnect from the RF670R. |

3 | If the connection was successfully terminated, "IDB_RF670R_RW".TCP_connect will be reset and "IDB_CALL_RF670R".Done will be set. |

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB3.DBDX 0.0</td>
<td>&quot;IDB_CALL_RF670R&quot;.Connect BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.1</td>
<td>&quot;IDB_CALL_RF670R&quot;.Read_Tag_IDs BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.2</td>
<td>&quot;IDB_CALL_RF670R&quot;.Write_Tag_Id BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.3</td>
<td>&quot;IDB_CALL_RF670R&quot;.Read_Memory BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.4</td>
<td>&quot;IDB_CALL_RF670R&quot;.Write_Memory BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.5</td>
<td>&quot;IDB_CALL_RF670R&quot;.Disconnect BOOL true</td>
</tr>
<tr>
<td>DB670.DBX 3352.0</td>
<td>&quot;IDB_RF670R_RW&quot;.TCP_connect BOOL false</td>
</tr>
<tr>
<td>DB3.DBDX 0.6</td>
<td>&quot;IDB_CALL_RF670R&quot;.Done BOOL true</td>
</tr>
</tbody>
</table>

(*1): A connection to the RF670R reader must already exist.
6 Further Information

This chapter shows you modifications to the program (e.g., changing the receive buffer length or modifications to the structure of DB RF_PARAMETER (UDT RF_VAR)).

In addition, we explain the function of the conversion blocks FB INT_TO_HEX (FB2) and FB HEX_TO_INT (FB3).

6.1 Modifications to the program

Changing the receive buffer length

If the total length of the response message is longer than the length of the receive buffer, you have to change the length of the receive buffer. The following table lists the steps necessary to make this change.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the “RF670R_RW” SCL code and change the receive buffer length.</td>
</tr>
<tr>
<td>2</td>
<td>Once again, save, compile and download the SCL code.</td>
</tr>
<tr>
<td>3</td>
<td>Open FB CALL_RF670R (FB1) and update the instances.</td>
</tr>
<tr>
<td></td>
<td>Once again, save the FB.</td>
</tr>
</tbody>
</table>
6 Further Information

6.1 Modifications to the program

4 Set the recompiled IDB_RF670R_RW instance data block to “Non Retain”.

5 Once again, download the entire project to your controller.

Modifications to the RF_VAR structure

If you want to edit UDT variables, e.g. the length of the “RF_VAR.REC_DATA” field, you have to follow the steps listed in the following table.

Table 6-2

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open UDT RF_VAR.</td>
</tr>
<tr>
<td>2</td>
<td>Edit the UDT variable, for example the length of the “RF_VAR.REC_DATA” field.</td>
</tr>
<tr>
<td>3</td>
<td>Open DB RF_PARAMETER in which the UDT is declared and update the instances.</td>
</tr>
</tbody>
</table>
6 Further Information

6.1 Modifications to the program

4 In DB RF_PARAMETER, once again enter the parameters of the TCP connection and all parameters necessary for RFID data processing and save DB RF_PARAMETER.

<table>
<thead>
<tr>
<th>No.</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>In DB RF_PARAMETER, once again enter the parameters of the TCP connection and all parameters necessary for RFID data processing and save DB RF_PARAMETER.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Initial value</th>
<th>Actual value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RF_VAR.IP Addr</td>
<td>DWORD</td>
<td>192.168.1.100</td>
<td>192.168.1.100</td>
</tr>
<tr>
<td>4</td>
<td>RF_VAR.DEVIC_ID</td>
<td>BYTE</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>6</td>
<td>RF_VAR.COM_ID</td>
<td>WORD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>RF_VAR.LOCAL_PORT</td>
<td>WORD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>RF_VAR.TAG_ID</td>
<td>STRING</td>
<td>&quot;&quot;</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>36</td>
<td>RF_VAR.new_ID</td>
<td>STRING</td>
<td>&quot;&quot;</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>62</td>
<td>RF_VAR.Source</td>
<td>STRING</td>
<td>&quot;Source 1&quot;</td>
<td>&quot;Source 1&quot;</td>
</tr>
<tr>
<td>318</td>
<td>RF_VAR.Value_bank</td>
<td>WORD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>325</td>
<td>RF_VAR.value_startAddress</td>
<td>WORD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>322</td>
<td>RF_VAR.value_decLength</td>
<td>WORD</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>324</td>
<td>RF_VAR.Duration</td>
<td>WORD</td>
<td>2077</td>
<td>2077</td>
</tr>
</tbody>
</table>

5 Once again, save and compile the “RF670R_RW” SCL code and update the instances in FB CALL_RF670R.

6 Set the recompiled IDB_RF670R_RW instance data block to “Non Retain”.

7 Once again, download the entire project to your controller.
6.2 Auxiliary blocks for converting a SIMATIC variable type for the RFID reader

All data that is written to or read from a transponder is available in hexadecimal character format.

If your raw data is available in a common SIMATIC numerical format, you first have to convert it to the format of the RF670R (hexadecimal character format).

The following blocks show this using the example of an integer.

6.2.1 FB INT_TO_HEX and FB HEX_TO_INT parameters

The following figure and table show the FB INT_TO_HEX call interface.

Figure 6-1

Table 6-3

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>BOOL</td>
<td>Enable input. Relevant only in FBD and LAD representation.</td>
</tr>
<tr>
<td>IN_INT</td>
<td>INT</td>
<td>Data that must be converted to hexadecimal characters. The data is fetched from DB In_Out_Data (see Figure 6-3).</td>
</tr>
<tr>
<td>OUT_HEX</td>
<td>ANY</td>
<td>Converted data to be written to the transponder.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INT</td>
<td>Byte data length. In this case: 2.</td>
</tr>
<tr>
<td>ENO</td>
<td>BOOL</td>
<td>Enable output. Relevant only in FBD and LAD representation.</td>
</tr>
</tbody>
</table>

The following figure and table show the FB HEX_TO_INT call interface.

Figure 6-2
6.2 Auxiliary blocks for converting a SIMATIC variable type for the RFID reader

Table 6-4

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>BOOL</td>
<td>Enable input. Relevant only in FBD and LAD representation.</td>
</tr>
<tr>
<td>IN_HEX</td>
<td>ARRAY [1..128] OF CHAR</td>
<td>Received data that must be converted to INT.</td>
</tr>
<tr>
<td>OUT_INT</td>
<td>INT</td>
<td>Converted data that is stored in DB In_Out_Data (see Figure 6-3).</td>
</tr>
<tr>
<td>ENO</td>
<td>BOOL</td>
<td>Enable output. Relevant only in FBD and LAD representation.</td>
</tr>
</tbody>
</table>

Figure 6-3 DB In_Out_Data

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>Struct</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>Int_IN</td>
<td>INTRFID data written to the transponder</td>
</tr>
<tr>
<td>2.0</td>
<td>Int_OUT</td>
<td>INTRFID data read from the transponder</td>
</tr>
<tr>
<td>4.0</td>
<td>END_STRUCT</td>
<td></td>
</tr>
</tbody>
</table>

Note

For more program details, the documented FB INT_TO_HEX (FB2) and FB HEX_TO_INT (FB3) SCL codes are available to you in the project (see \1).
6.2 Auxiliary blocks for converting a SIMATIC variable type for the RFID reader

### 6.2.2 Call of FB INT_TO_HEX and FB HEX_TO_INT in OB1

If you write data of the INT type from a data source to a transponder or if you want to store the data read from a transponder in this data source as INT, you have to call FB INT_TO_HEX and FB HEX_TO_INT.

Figure 6-4
7 References

7.1 References

This list is by no means complete and only presents a selection of related references.

Table 7-1

<table>
<thead>
<tr>
<th>Topic</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>/1/</td>
<td>STEP7 SIMATIC S7-300/400 Automating with STEP7 in STL and SCL Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-412-5</td>
</tr>
<tr>
<td>/2/</td>
<td>STEP7 SIMATIC S7-300/400 Automating with STEP 7 in LAD and FBD Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-410-1</td>
</tr>
<tr>
<td>/3/</td>
<td>STEP7 SIMATIC S7-300 Automating with SIMATIC S7-300 inside TIA Portal Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-382-1</td>
</tr>
</tbody>
</table>

7.2 Internet links

This list is by no means complete and only provides a selection of useful information.

Table 7-2

<table>
<thead>
<tr>
<th>Topic</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>/2\</td>
<td>Siemens Industry Online Support <a href="http://support.automation.siemens.com">http://support.automation.siemens.com</a></td>
</tr>
<tr>
<td>/3\</td>
<td>Overview of communication services supported by PN-CPU's <a href="http://support.automation.siemens.com/WW/view/en/18909487">http://support.automation.siemens.com/WW/view/en/18909487</a></td>
</tr>
</tbody>
</table>
## 7.2 Internet links

<table>
<thead>
<tr>
<th>Topic</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function Manual</td>
</tr>
</tbody>
</table>

## 8 History

### Table 8-1

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>06/2012</td>
<td>First edition</td>
</tr>
<tr>
<td></td>
<td>08/2012</td>
<td>Note in chapter 2.4 has been changed</td>
</tr>
<tr>
<td>V1.1</td>
<td>04/2013</td>
<td>Note in chapter 2.1 has been inserted and chapter 3.3 has been changed</td>
</tr>
</tbody>
</table>