WinAC FileServer

Operator manual

V 1.2.3 • March 2012

Applikationen & Tools

Answers for industry.

SIEMENS

Industry Automation and Drives Technologies Service & Support Portal

This article is taken from the Service Portal of Siemens AG, Industry Automation and Drives Technologies. The following link takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/55422031

If you have any questions concerning this document please e-mail us to the following address:

online-support.automation@siemens.com

applications.aud.koe.nrh.rd@siemens.com

S

SIMATIC
WinAC FileServer

Operator manual

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 used correctly. These application examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we 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 – 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”). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Table of Contents

1 Automation Task

1.1 Overview

Introduction

It is not possible to save or load files into data blocks. This functionality is provided by the **WinAC FileServer** driver. Various file types are supported: Binary, ASCII, XML, SCV and Windows INI file.

1.1 Needed knowledge

To understand this document the following knowledge needed:

- SIMATIC WinAC RTX 2010
- SIMATIC Manager STEP7 V5.5

  or
- STEP7 V11 (TIA Portal)

1.2 Required Hardware and Software Components

The application was generated with the following components:

Hardware components

- Simatic Microbox IPC 427C (1,2 GHz, 4 GB RAM, 4 GB CF-Card) with Windows XP embedded SP3

Standard software components

- SIMATIC WinAC RTX 2010
- SIMATIC Manager STEP 7 V5.5

Sample files and projects

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

Table 1-1 Included files

Component	Note
Documentation \ WinAC-FileServer_Doku_v11_DE.pdf Documentation \ WinAC-FileServer_Doc_v11_EN.pdf	This documentation in German and English
Driver \ WinAcFileServer.dll	The WinAC driver
Driver \ setup.bat	Setup batch for the driver
S7ClassicExample \ FileServer.zip	Step7 example project including all WinAC function blocks of the driver
S7V11-TIA-PortalExample	Step7 example project including all WinAC function blocks of the driver
ConfigDbCreator \	Tool for creating configuration data blocks

2 Overview

2.1 Functional range

The WinAC FileServer provides functions to store and read data block in a structured format. Several file types are supported (XML, ASCII, Binär, CSV, INI).

The driver provides the following functions:

- Access several files at same time
- Writing/Reading in structured format (XML, CSV, etc.)
- Continuous writing (e.g. measured values in “append” mode)
- File status (size, date, etc.)
- File handling (delete, copy, etc.)

Note The driver reads / writes data blocks as a whole by default

The format of the XML file is fixed. The driver gives the possibility to write and read data blocks among other things in XML format. It is not possible to parse any user-defined XML files.

2.2 Version of the driver

Determine driver under Windows OS

The driver DLL is located in the system32 directory, e.g.

C:\Windows\system32\WinAcFileServer.dll

You can identify the version of the driver RTDLL in the file properties (Windows explorer → right click → properties)

Figure 2-1 Properties of the driver DLL

Check driver version in Step7 project

In the instance DB of FBFileInit the version of the DLL is stored, too:

tOdkIf.dwDllVersion version of the driver DLL

3 Driver supported functionality

3.1 General overview

The driver consists of two parts:

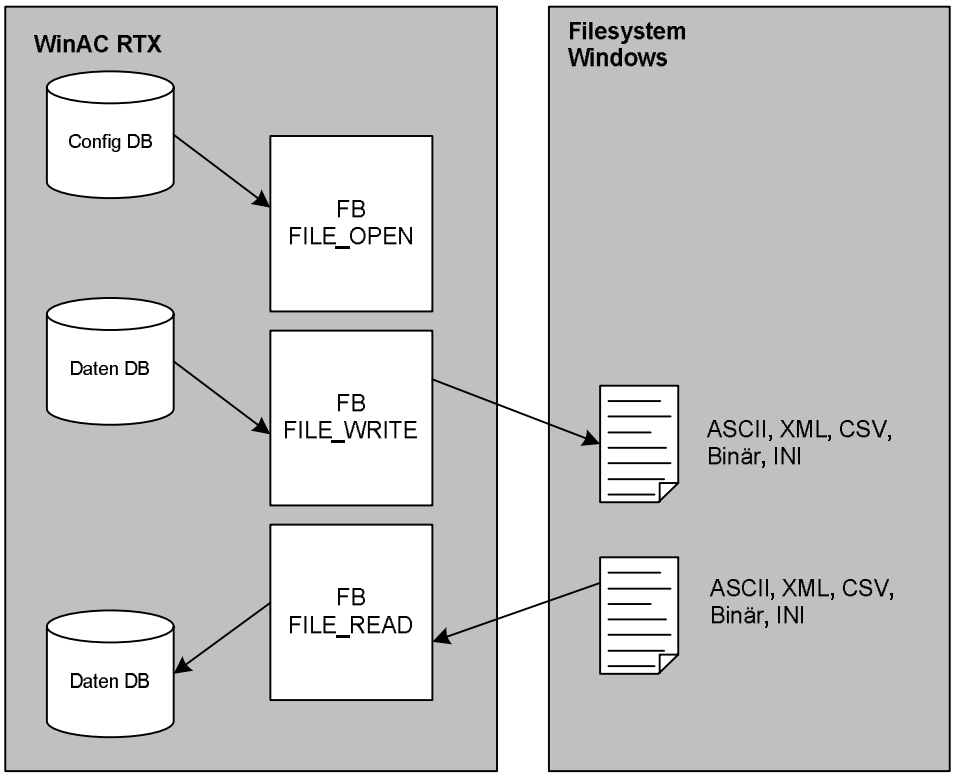
- The Step7 function blocks
- The driver DLL

Thus the driver DLL must be installed on the runtime system. The Step7 function blocks are used in the Simatic Manager on the Engineering station.

To be able to read/write data blocks in structured format, the driver has to know about the internal structure of the data block (data types, arrays etc.).

The needed information about the composition of the data block is stored in a “config data block”. The advantage is: all needed information is part of the Step7 project. If the information would be stored in some kind of INI file, the WinAC projecting would consist of two parts. The INI file has to be copied to the runtime system.

Figure 3-1 Structure of the solution



The needed “config data block” can be created by the provided tool “Config DB Creator”.

Note

A separate FB FILE_CLOSE is not needed.

After writing a file it is closed immediately.

If another data block has to be written to a different file, just call the FB FBFileOpen again with modified parameters.

Note

The reading and writing was developed for handling **one** file. E.g. a data of one file must fit in one data block.

3.2 Auxiliary functions for file handling

The driver includes function blocks for file handling. They provide the following functions:

- Delete files
- Rename files
- Check status (size, date)
- Copy files

3.3 Storing information about data block's structure

The driver needs information about the internal build-up of the written / read data block. Only with this information a structured file format is possible.

There are two possibilities:

- Some kind of INI file
- Separate config data block

The WinAC FileServer uses the solution with a separate config data block. Thus the complete project is stored in the Simatic Mager project. Project download is done by the Simatic Manager, only. No additional copying of some INI file is needed.

This solution has a drawback: the size of the DB is limited by the number of structure information stored in the config DB.

Attention

The present version of the driver can handle maximum 4.000 items for every file.

The Config data block

The config data block contains information needed for handling data in structured format.

The principle build-up is shown below:

Table 3-1 Example of data block to read / write (STL source)

```
STRUCT
  Par_Kp INT
  Par_Tn INT
  Heater STRUCT
        Ht_Topt INT
        Ht_Trcp INT
  END_STRUCT
  Par_Ti INT
END_STRUCT
```

Table 3-2 Matching config data block (STL source)

```
// Header
CONFIG_DB_VERSION : INT := 1; //Version of ConfigDB
CSV_SEPERATOR : CHAR := ';'; //CSV-Seperator
WITH_HEADER : BOOL := TRUE; //Use header for files
RESERVE : ARRAY  [1 .. 96 ] OF BYTE ;
// Type information
CFG_DATA_TYP_1 : BYTE := B#16#22; //INT
CFG_DATA_NAM_1 : STRING [6] := 'Par_Kp';
CFG_DATA_TYP_2 : BYTE := B#16#22; //INT
CFG_DATA_NAM_2 : STRING [6] := 'Par_Tn';
CFG_DATA_TYP_3 : BYTE := B#16#70; //STRUCT_START
CFG_DATA_NAM_3 : STRING [7] := 'Heater';
CFG_DATA_TYP_4 : BYTE := B#16#22; //INT
CFG_DATA_NAM_4 : STRING [7] := 'Ht_Topt';
CFG_DATA_TYP_5 : BYTE := B#16#22; //INT
CFG_DATA_NAM_5 : STRING [7] := 'hat_Trcp';
CFG_DATA_TYP_6 : BYTE := B#16#71; //STRUCT_END
CFG_DATA_NAM_6 : STRING [0] := ''; //Struct ends here
CFG_DATA_TYP_7 : BYTE := B#16#22; //INT
CFG_DATA_NAM_7 : STRING [6] := 'Par_Ki';
CFG_DATA_TYP_8 : BYTE := B#16#AA; //End of ConfigDB
CFG_DATA_NAM_8 : STRING [0] := '';
```

To store the variable names only the needed space is used for the SIMATIC strings.

Coding of Step7 data types

All available S7 data types have to be stored.

Additional sizes of Arrays or maximum length of strings have to be known. This information is stored in the config DB, too.

For arrays the start index is stored because it can be different than 0

Table 3-3 Coding of S7 data types in config DB

```
HEX    DEC   Description
----------------------------------------
00h -   0d – not used
01h -   1d - Type BOOL
10h -  16d - reserve 1 byte
11h -  17d - Type BYTE
12h -  18d - Type CHAR
20h -  32d - reserve 2 byte
21h -  33d - Type WORD
22h -  34d - Type INT
23h -  35d - Type S5TIME
24h -  36d - Type DATE
40h -  64d - reserve 4 byte
41h -  65d - Type DWORD
42h -  66d - Type DINT
43h -  67d - Type REAL
44h -  68d - Type TIME
45h -  69d - Type TIME_OF_DAY
50h -  80d - Type STRING
51h -  81d -      max. len. of string
60h -  96d - Type ARRAY
61h -  97d -      start index of array
62h -  98d -      length of array
63h -  99d -      end of array
70h - 112d - Type STRUCT (start of struct)
71h - 113d -      end of struct
72h - 114d - Type UDT (start of UDT)
73h - 115d -      end of UDT
80h - 128d - reserve 8 bytes
81h - 129d - Type DATE_AND_TIME
AAh - 170d - 'END of DB'
```

Config data block header

Some file related information is stored in the header of the config data block like the separator for the CSV file.

Table 3-4 Parameters of config data block header

Parameter	Typ	Beschreibung
CONFIG_DB_VERSION	INT	Version of Config DBs *1)
WITH_HEADER	BOOL	Create Header in data file *2)
CSV_SEPERATOR	CHAR	Separator for CSV files
RESERVE		reserved

*1) The config data block version number is needed to guarantee a matching between configuration an WinAC FileServer version. If in future versions more information is stored in the header, it will be marked by a changed version number.

*2) Creating a header is only used for CSV files.

3.4 Tool for creating configuration data block (Config DB Creator)

This tool supports in creating a matching config data block. It creates a config data block for a selected data block.

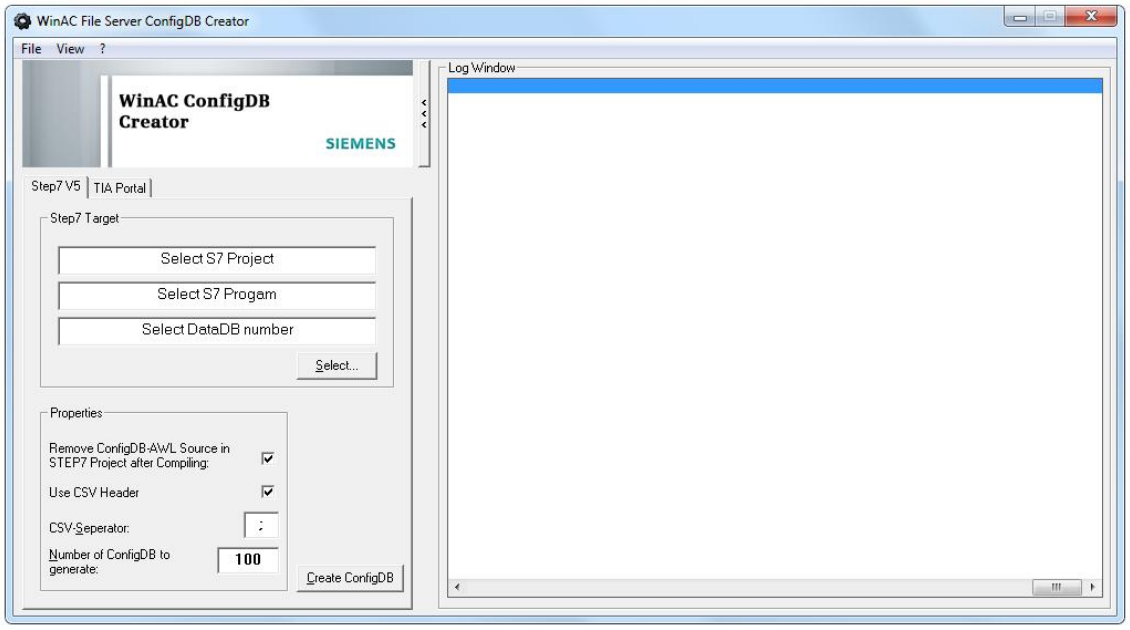
Note The "Config DB Creator" supports analyzing of global data blocks, only. It is not possible to analyze instance DBs because the sources of such blocks does not include the needed data type information.

The user selects the Step7 project, the folder and the data block. After that the tool analyses the data block and generates the matching config data block.

3.4.1 Usage of ConfigDBCreator with Step7 Classic (V5.5)

Note The "Config DB Creator" needs an installed Simatic Manager V5.5 for operation.

Figure 3-2 GUI of the Config DB Creator



On the right side a log window is shown. It can be hidden by pressing '<<<'. With the menu "View / Clear log" the log window can be cleared.

To change the Step7 target (project / folder / data block) the button 'Select' has to be pressed.

The 'Properties' area contains some additional settings, like the CSV separator. The setting "Use header" is used for CSV file type only! It the checkbox is activated, a first line with the variable names is written. When reading a file containing a header line this first line is skipped, because it contains the header only.

By pressing 'Create ConfigDB' the tool generates the new data block in the selected Step7 project.

3.4.2 Using Step7 V11 (TIA-Portal)

The TIA Portal V11 does not provide an application programming interface (API) for external applications. Thus the „DB Config Creator“ can’t read and automatically create new DBs in a Step7 Project. One has to use the Export/Import functionality of DBs of the TIA Portal. The following steps are needed:

- Generate source for the data DB
- Generate sources for the used UDTs (if UDTs are used in the data DB).
- Use “Config DB Creator” to build the source for the Config DB
- Import the source of the Config DB in the TIA Portal

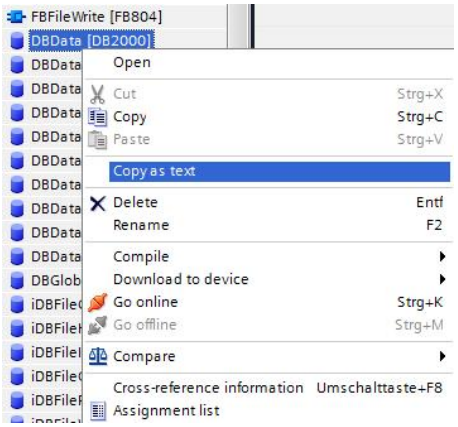
This approach is described by the following section.

Generate sources in TIA Portal

To create a config data block in the TIA-Portal, the source of the DB is needed. This source is created by the following steps:

- Right mouse click on the data block
- Select *„copy as text“ (Figure 3-3)* in the context menu
- Open a text editor (e.g. Notepad) and paste the source text
- Save the file.

Figure 3-3 „copy as text“ in context menu



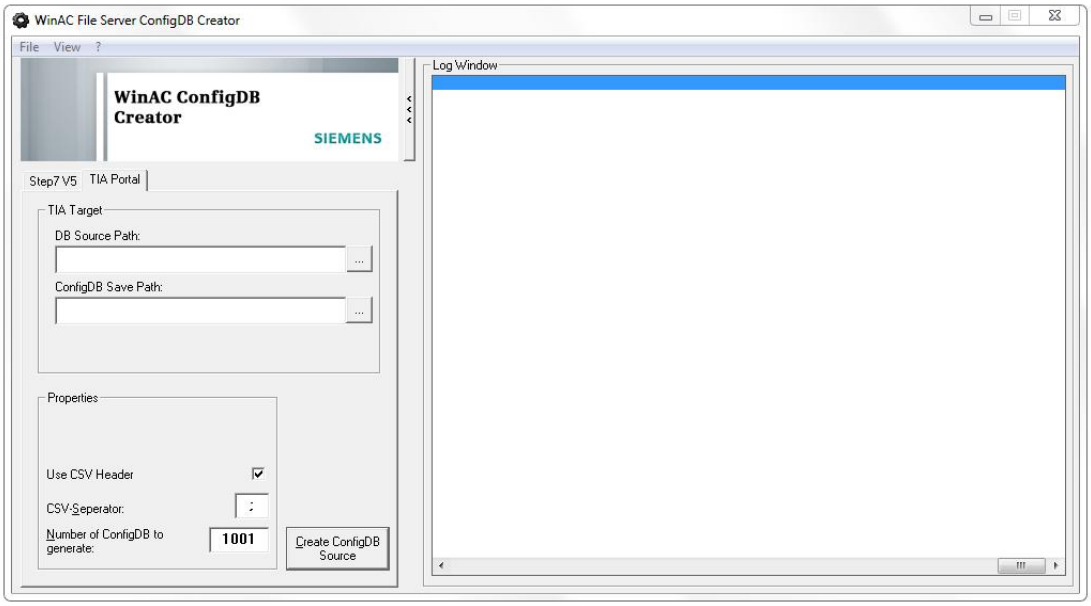
NOTE Beside the data block to read/write with WinAC FileServer, additional the sources for used UDTs are needed, too.

- Store sources of UDTs in the same directory where the DBs was saved.
  The file names must match following syntax:
  "<Name of UDT>_SRC_AWL".

Analyzing and creating of config DB with “Config DB Creator”

- Start the ConfigDB Creator and click on the tab *„TIA Portal“*.
- Select the file in the text field *“DB Source File“*.
- Define the path for storing the *“config DB”*
- Assign a number for the config DB.
- Click on *„Create ConfigDB*“.

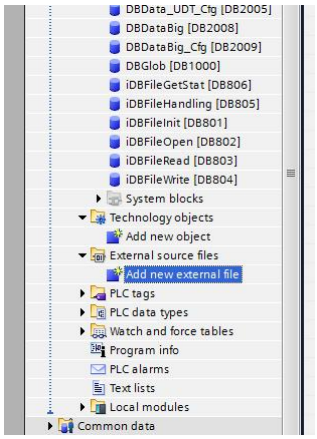
Figure 3-4 „WinAC ConfigDB Creator“ GUI - TIA Portal



Import source of “Config DB” into TIA portal project

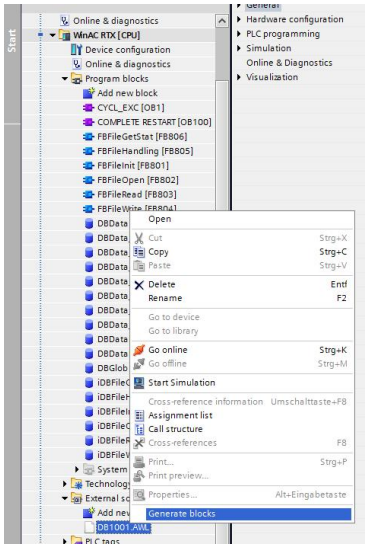
- In the folder *„external source“* you can find the submenu *„add external files"*. Double clicking on the item will open a new dialog. In this dialog you can select the text file.

Figure 3-4 Projectview in the TIA Portal („add external files“)



- Now you see the external source in the folder *“external source files”*.
- Right click on the file opens the submenu.
- Clicking on *„generate block“* opens a new dialog showing the actual state.

Figure 3-5 *„generate block“* submenu



- After generating the data block you find the block in the folder *“Program blocks”*

NOTE If the external source of the Config DB is changed, a new manually import is needed in the TIA Portal. An updated external source file is not automatically updated in the TIA portal!

3.5 Supported File Format

3.5.1 Binary file

Write:	YES
Append mode:	NO
Read:	YES
File Header:	NO
Repeat Counter:	NO [*1)]

This file format is the right solution, if a minimum space is required or if the internal data should be hidden.

If a file is read into a data block, it has to be guaranteed the file was generated from a data block with exact same structure.

[*1)] A data block is written complete to the file or read complete from the file. That's why a repeat counter for multiple read/write operations makes no sense.

3.5.2 CSV – Comma Separated Values

Write:	YES
Append mode:	YES
Read:	YES (not continuously data)
File Header:	YES (parameter) [*2)]
Repeat Counter:	YES [*3)]

When using CSV the data is stored "flat", i.e. information about arrays or structures inside the data block are lost.

Some parameters are specific for CSV files:

- Separator (tabulator, space, comma, etc.)
- Header line with variable's names

NOTE The separator character can be defined by the user. Pay attention the used character is not contained in the user data.

[*2)] Function of "with header line"

If the property "with header line" is activated, it causes the following:

- When a file is written, a first header line with all variable names is written before the data.
- When a file is read, the first line is skipped, because it should include the header only and no data.

Table 3-5 Example for CSV file with header

```
Par_Kp; Par_Tn; Par_Ti
1; 3.4; 7
```

Table 3-6 Example for CSV file without header

```
1; 3.4; 7
```

*3) Repeat Counter / Multiple read/write operations

In some applications e.g. some measure data is collected in one data block (e.g. time, value 1, value 2, time, value 1, value 2). The goal is to create a CSV file like this:

Table 3-7 Example for CSV file multiple read/write (usage of repeat counter)

```
time; value 1; value 2
TOD#10:23:29.123; 3.4; 7.7
TOD#10:24:31.123; 3.6; 8.6
TOD#10:25:28.123; 3.7; 9.8
```

You could create a file like this with multiple call of FB WRITE_FILE with APPEND flag is set.

But if all data is stored in one data block it is more effective to do it with one call. When using the repeat-counter you can write all lines with one call of FB WRITE_FILE. For the example above the repeat-counter must have the value of 3.

The ‘repeat-counter’ can be used for reading, too.

Note When the ‘repeat-counter’ is used, keep in mind the data must fit in one data block

3.5.3 ASCII – American Standards Committee for Information Interchange

Write:	YES
Append mode:	YES
Read:	YES (not continuously data)
File Header:	NO
Repeat Counter:	NO

When using ASCII a header (line) is not needed because the variable's name is written in front of every value.

Table 3-8 Example for ASCII file in append mode

```
Par_Kp = 1
Par_Tn = 3.4
Par_Ti = 7

Par_Kp = 2
Par_Tn = 3.6
Par_Ti = 9
```

Table 3-9 Example for ASCII file with array of struct

```
StructArray[1].Flag21=FALSE
StructArray[1].Flag22=FALSE
StructArray[2].Flag21=FALSE
StructArray[2].Flag22=FALSE
```

3.5.4 XML - Extensible Markup Language

Write:	YES
Append mode:	YES
Read:	YES (not continuously data)
File Header:	NO
Repeat Counter:	NO

With file type XML the data block can be written into a XML file including all structure and type information.

The format of the XML file is fixed. It is not possible to parse any user-defined XML files.

The following example shows an example of a data block including a array and a structure.

Table 3-10 Example for XML file

```
<?xml version="1.0"?>
<!--  generated by WinAC's file server, 25.03.2010 -->
<winac_data>
  <element name="Par_Kp" Type="INT" value="12"/>
  <element name="Par_Tn" Type="INT" value="24"/>
  <element name="Heater" Type="STRUCT">
    <element name="Ht_Topt" Type="INT" value="56"/>
    <element name="Ht_Trcp" Type="INT" value="78"/ >
  </element>
  <element name="Par_Ti" Type="INT" value="91"/>
  <array name="RefPoints Type="INT">
    <arrayitem index="1" value="3"/>
    <arrayitem index="2" value="8"/>
  </array>
  <array name="StructArray" type="STRUCT">
    <arrayitem index="1">
      <element name="Flag21" type="BOOL" value="FALSE"/>
      <element name="Flag22" type="BOOL" value="true"/>
    </arrayitem>
    <arrayitem index="2">
      <element name="Flag21" type="BOOL" value="FALSE"/>
      <element name="Flag22" type="BOOL" value="true"/>
    </arrayitem>
  </array>
</winac_data>
```

When the XML file is created, the creation date is automatically added in the header area (see example above).

Note The structure of the XML file is fixed (see example). It is not possible to read a XML file with any XML format.

3.5.5 INI – Windows initialization file

Write:	YES
Append mode:	NO
Read:	YES
File Header:	NO
Repeat Counter:	NO

Note When using INI files, a specific structure of the data block is needed.

This file type is special: it requires a specific structure of the data block. Only in this way it is possible to map a data block to a INI file and vice versa map a INI file to a data block.

Table 3-11 Example for Windows INI file

```
[Parameter]
KP=6
TN=5.3
[HMI]
NUTZER=PAUL
LEVEL=5
```

Table 3-12 Matching data block to INI file above

```
STRUCT

  Parameter : STRUCT
    KP : INT  := 6;
    TN : REAL  := 5.300000e+000;
  END_STRUCT ;

  HMI : STRUCT
    Nutzer : STRING  [254] := 'Paul';
    Level : INT  := 5;
  END_STRUCT ;

END_STRUCT ;
```

I.e. the sections of the INI file are represented by STRUCTs (or UDTs) in the data block. Nested structures are not supported.

If an INI file is written all information from the data block is inserted or updated in the INI file. If there is additional information in the INI file, it stays untouched.

If an INI file is read only the needed information is retrieved. Any additional information in the INI file is ignored.

3.6 Supported Step7 data types

3.6.1 Basic types

All Step7 data types are supported. The WinAC FileServer has to write and read the data as strings (except for binary files).

The following table shows the supported Step7 data types.

Table 3-13 Formatting of supported Step7 data types

Data type	Formats WRITE	Valid formats READ
BOOL	TRUE / FALSE	true false TRUE FALSE 0 1
BYTE	B#16#AB	B#16#AB 0xAB
CHAR	a	' ' (Leerzeichen) – '€' (EUR)
WORD	W#16#ABCD	W#16#ABCD 0xABCD
DWORD	DW#16#ABCDEF01	DW#16#ABCDEF01 0xABCDEF01
INT	2	2
DINT	L#83	L#83 83
REAL	1.235789e+000	0.03 5.000000e+001
STRING	abcde	abcde
S5TIME *1)	S5T#1M39S900MS	S5T#1M39S900MS 1M39S900MS
TIME	T#4d20h31m23s647ms	T#4d20h31m23s647ms 4d20h31m23s647ms
TIME_OF_DAY	TOD#12:59:59.999	TOD#12:59:59.999 12:59:59.999
DATE	D#1990-02-01	D#1990-01-01 1990-01-01
DATE_AND_TIME	DT#10-9-29-10:54:21.123	DT#10-9-29-10:54:21.123 10-9-29-10:54:21.123

Note The data types ANY and POINTER are not supported!

*1) Depending on the value of the time the data type S5TIME internally utilizes different time bases: 10ms 100ms 1s 10s. Portions smaller than the used time base are lost.

3.6.2 Structured data types

STRUCT and UDTs are supported.

UDT is processed like STRUCT. In the XML file is not difference between using a STRUCT or an UDT in the data block.

Array inside arrays is supported.

Multi-dimensional arrays are **NOT** supported!

4 WinAC function blocks (FB)

The WinAC FileServer provides a number of function blocks for access all functionality. The numbers of the function block can be changed by the user.

FB801 – FBFileInit
FB802 – FBFileOpen
FB803 – FBFileRead
FB803 – FBFileWrite
FB805 – FBFileHandling
FB806 – FBFileGetStat

Note The driver function blocks are implemented in SCL (source included). For the usage of the WinAC FileServer driver is SCL not needed!

4.1 FBFileInit

This block initializes the WinAC FileServer. It has to be called one-time before any other call of FileServer function blocks.

Table 4-1 Parameter of FBs FBFileInit

Parameter	In/ Out	Typ	Description
ERROR	Out	Bool	Error
STATUS	Out	WORD	Status information
ODK_REF	Out	WORD	Reference to driver *1)

Note *1) The value ODK_REF hast to provided to all other function blocks of the FileServer.

Additional information in instance DB of FBFileInit

Additional to the output parameters some information is stored in the instance data block of the function block:

Table 4-2 Information in instance data block of FBFileInit

Name	In/ Out	Description
CIF.DLL_VERSION	In	Version of driver DLL

Coding of DLL version

The DLL version is coded hexadecimal. The last sign of the DWORD is used for mark Debug and Release version:

D – Debug-Version

A – Release-Version

Figure 4-1 Examples for DLL versions in the instance DB

```
"iDB_FILE_INIT".iOdkIf.dwDllVersion   HEX   DW#16#0001000D
                                                \  /|
                                                 \/ +- Debug
                                                 +---- V 1.0.0.0


"iDB_FILE_INIT".iOdkIf.dwDllVersion   HEX   DW#16#0001100A
                                                 \  /|
                                                  \/ +- Release
                                                  +---- V 1.1.0.0
```

Note The data in the instance data block is valid after the first call of INIT function block!

4.2 FBFileOpen

This block opens a file. The structure information of the data block to read/write is provided

The function block works asynchronously. A rising edge of the REQ input starts the processing, a signaled DONE shows the end of processing.

The parameter ID is used for identifying a specific file. Up to 32 files are supported at same time.

Note If a specific ID should be used for a different file, the FBFileOpen has to be called witch changed parameters.

Table 4-3 Parameter of FBFileOpen

Parameter	In/ Out	Typ	Description
ODK_REF	In	WORD	Reference to the driver (see FBFileInit)
ID	In	INT	File identifier 0..31
FILENAME	In	STRING	File name (incl. Path)
FORMAT	In	BYTE	File format 1 - BIN, 2 - CSV, 3 - ASCII, 4 - XML, 5 – INI
CFG_DB	In	ANY	ANY-Pointer to data block with structural information („Config DB“) *1)
RESET	In	BOOL	Reset internal variables (has to be se tone time before first usage)
APPEND	In	BOOL	True – append mode False – overwrite mode
REQ	In	BOOL	Rising edge opens file
BUSY	Out	BOOL	Command is running
DONE	Out	BOOL	Command finished without error
ERROR	Out	BOOL	Error
STATUS	Out	WORD	Status information

*1) When using file type “binary” the parameter CFG_DB can stay empty because no configuration information is needed for binary.

4.2.1 Relationship between REQ, BUSY, DONE and ERROR

A rising edge of REQ starts the execution of the command. During the execution the BUSY flag is high. After finishing the command either DONE or ERROR becomes high (for minimum one FB call). If REQ is reset also DONE or ERROR respectively fall to low in the next call.

The following figures show the timing behaviour:

Figure 4-2 Time chart: REQ high, processing without error

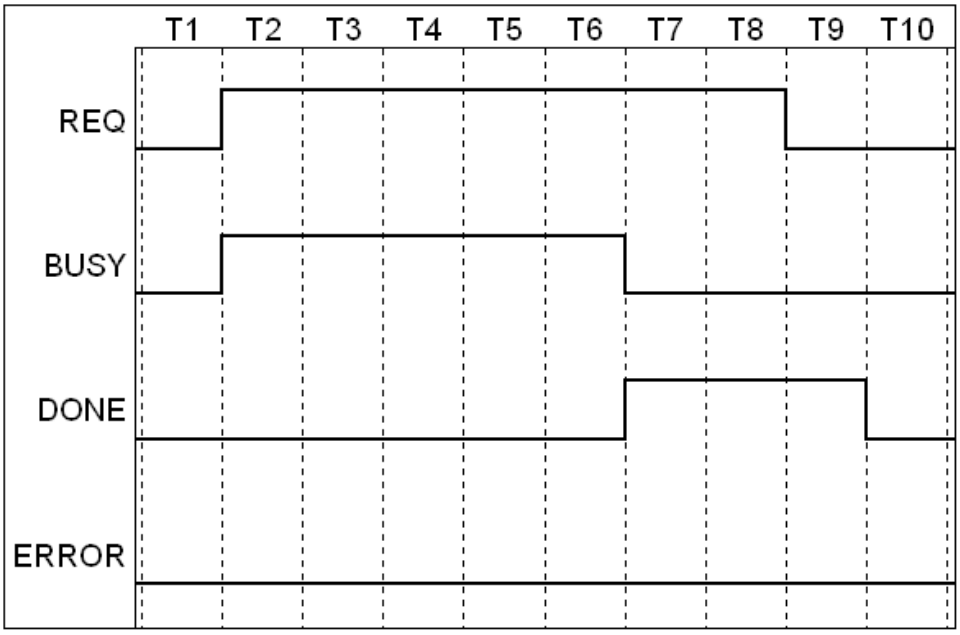


Figure 4-3 Time chart: REQ high, processing with error

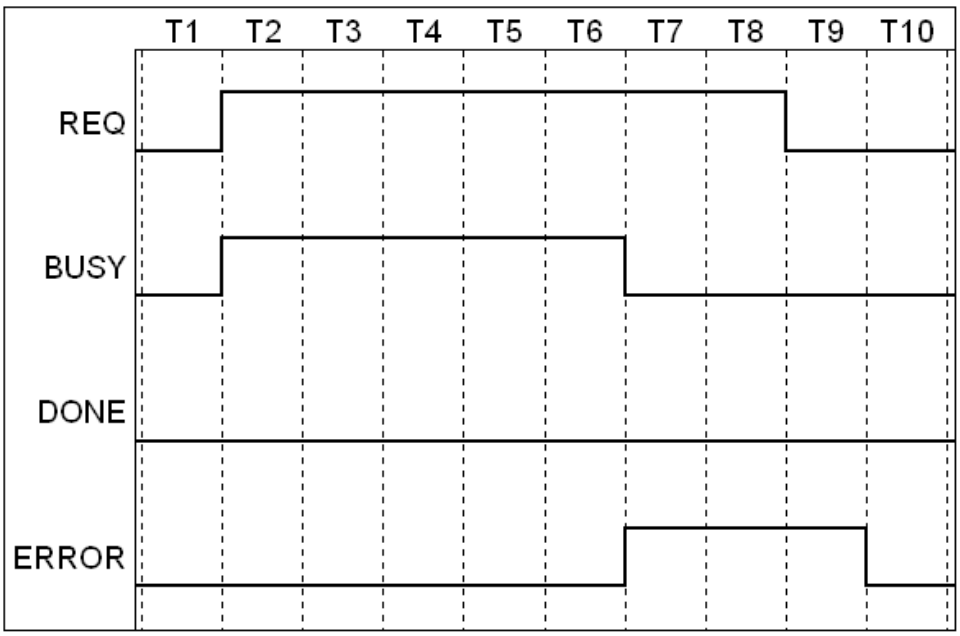


Figure 4-4 Time chart: REQ short pulse, processing without error

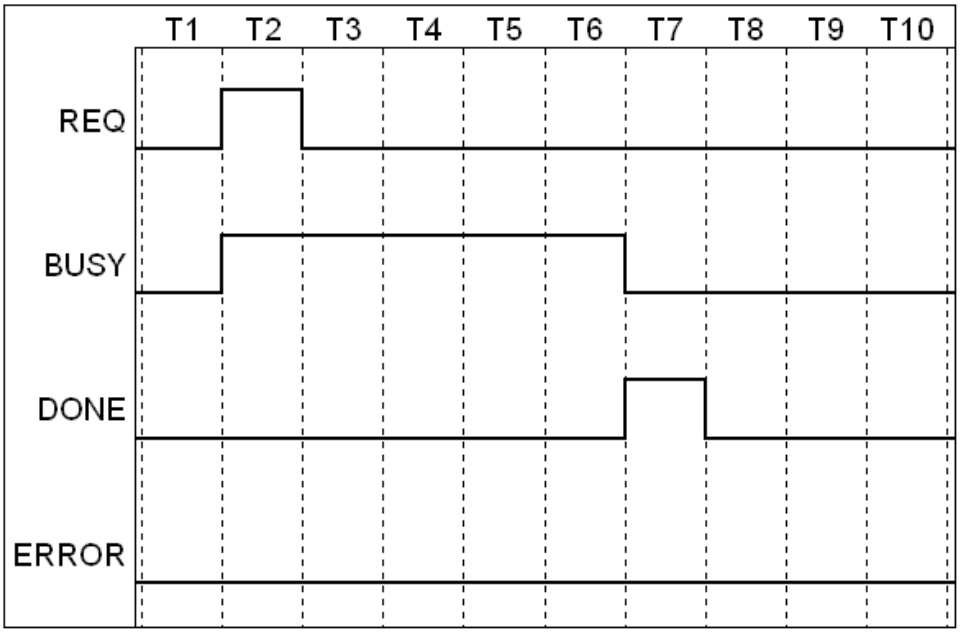
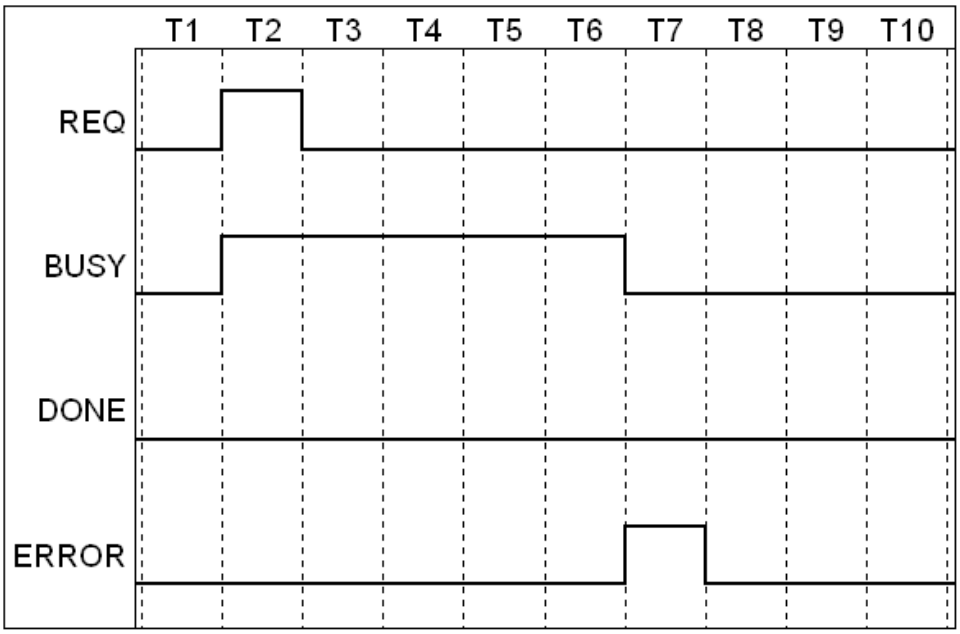


Figure 4-5 Time chart: REQ short pulse, processing with error



4.3 FBFileWrite

This block writes data to a file. An ANY pointer to the data block with the data is provided. The internal structure of the data block is known because of the call of FBFileOpen call before.

The function block works asynchronously. A rising edge of the REQ input starts the processing, a signaled DONE shows the end of processing.

The parameter ID is used for identifying a specific file. Up to 32 files are supported at same time.

Table 4-4 Parameter of FBFileWrite

Parameter	In/ Out	Typ	Description
ODK_REF	In	WORD	Reference to the driver (see FBFileInit)
ID	In	INT	File identifier 0..31
REPEAT_CNT	In	INT	Repeat counter: How often contains the data block (DATA_DB) the structure defined in CFG_DB (FBFileOpen)
DATA_DB	In	ANY	ANY pointer to data block to be written
RESET	In	BOOL	Reset internal variables (has to be se tone time before first usage)
REQ	In	BOOL	Rising edge opens file
BUSY	Out	BOOL	Command is running
DONE	Out	BOOL	Command finished without error
ERROR	Out	BOOL	Error
STATUS	Out	WORD	Status information

The time chart for REQ, BUSY, DONE and ERROR is described in chapter 4.2.1 on page 29.

NOTICE **The path where the file should be written must exist before calling the function FBFileWrite.**

4.4 FBFileRead

This block reads data from a file into a data block. An ANY pointer to the data block is provided. The internal structure of the data block is known because of the call of FBFileOpen call before.

The function block works asynchronously. A rising edge of the REQ input starts the processing, a signaled DONE shows the end of processing.

The parameter ID is used for identifying a specific file. Up to 32 files are supported at same time.

Table 4-5 Parameter of FBFileRead

Parameter	In/ Out	Typ	Description
ODK_REF	In	WORD	Reference to the driver (see FBFileInit)
ID	In	INT	File identifier 0..31
REPEAT_CNT	In	INT	Repeat counter: How often contains the data block (DATA_DB) the structure defined in CFG_DB (FBFileOpen)
DATA_DB	In	ANY	ANY pointer to data block to be read
RESET	In	BOOL	Reset internal variables (has to be se tone time before first usage)
REQ	In	BOOL	Rising edge opens file
BUSY	Out	BOOL	Command is running
DONE	Out	BOOL	Command finished without error
ERROR	Out	BOOL	Error
STATUS	Out	WORD	Status information

The time chart for REQ, BUSY, DONE and ERROR is described in chapter 4.2.1 on page 29.

4.5 FBFileHandling

With this function block the following functions are provided:

- Copy file
- Rename file
- Delete file
- Move file

Table 4-6 Parameter of FBFileHandling

Parameter	In/ Out	Typ	Description
ODK_REF	In	WORD	Reference to the driver (see FBFileInit)
FILE_SRC	In	String	Full path / filename before
FILE_DST	In	String	Full path / filename after
RESET	In	Bool	Reset all internal variables
RENAME	In	Bool	1 – rename file
MOVE	In	Bool	1 – move file
COYP	In	Bool	1 – copy file
DELETESRC	In	Bool	1 – Delete source file
OVERWRITE	In	Bool	1 –Overwrite if exists (only valid for COPY)
REQ	In	BOOL	Rising edge opens file
BUSY	Out	BOOL	Command is running
DONE	Out	BOOL	Command finished without error
ERROR	Out	BOOL	Error
STATUS	Out	WORD	Status information

The time chart for REQ, BUSY, DONE and ERROR is described in chapter 4.2.1 on page 29.

4.6 FBFileGetStat

With this function block you can get file status (size, date, etc.).

Table 4-7 Parameter of FBFileGetStat

Parameter	In/ Out	Typ	Description
ODK_REF	In	WORD	Reference to the driver (see FBFileInit)
FILE_NAME	In	String	Full path / filename
RESET	In	Bool	Reset all internal variables
REQ	In	BOOL	Rising edge opens file
BUSY	Out	BOOL	Command is running
DONE	Out	BOOL	Command finished without error
ERROR	Out	BOOL	Error
STATUS	Out	WORD	Status information
DISC_SPACE	Out	Dword	Free disk space
FILE_SIZE	Out	Dword	File size
FILE_TIME	Out	Date_and_ Time	Last change of file
FILE_ATTRIB	Out	Dword	File attributes Bit 0 - READONLY 0x00000001 Bit 1 - HIDDEN 0x00000002 Bit 2 - SYSTEM 0x00000004 Bit 4 - DIRECTORY 0x00000010 Bit 5 - ARCHIVE 0x00000020 Bit 6 - ENCRYPTED 0x00000040 Bit 7 - NORMAL 0x00000080 Bit 8 - TEMPORARY 0x00000100 Bit 9 - SPARSE_FILE 0x00000200 Bit 10 - REPARSE_POINT 0x00000400 Bit 11 - COMPRESSED 0x00000800 Bit 12 - OFFLINE 0x00001000

The time chart for REQ, BUSY, DONE and ERROR is described in chapter 4.2.1 on page 29.

5 Installation

5.1 Quick-start

Run-time system

- Install the driver DLL on the run-time system with the **setup.bat**

Engineering system Step7 V5.5

- Retrive and open the archived Step7 project with the Simatic Manager
- Copy all needed driver function blocks in your project (incl. SFBs)
- Run the ConfigDBCreator.exe for creating the needed configuration DB
- Call FBFileInit once before using any other FB from the driver
- Call FBFileOpen with right file name
  (positive edge opens file)
- Call FBFileRead / FBFileWrite for reading / writing
  (positive edge triggers function)

Engineering system TIA Portal V11

- Retrive and open the archived Step7 project with the Simatic Manager
- Copy all needed driver function blocks in your project
  Used SIMATIC PLC must be an WinAC. In other case the driver's function blocks are marked as faulty.
- Run the ConfigDBCreator.exe for creating the needed configuration DB
  (manual import / export of sources is needed)
- Call FBFileInit once before using any other FB from the driver
- Call FBFileOpen with right file name
  (positive edge opens file)
- Call FBFileRead / FBFileWrite for reading / writing
  (positive edge triggers function)

5.2 Installation WinAC driver on runtime system

The installation of the WinAC driver is limited to the copying of the driver DLL to the system32 directory. For that purpose there is a batch file **setup.bat**.

Installation under Windows XP (embedded)

Under Windows XP one can start the **setup.bat**. This works from USB stick, too.

Installation under Windows 7

For copying a file to system32 Administrator privileges are needed. Thus the **setup.bat** has to be started as Administrator (right click – Run as Administrator). The Windows 7 UAC has to be confirmed with “Yes”.

This works from USB stick, too.

5.3 Installation WinAC driver on engineering system with Step7 classic

On the engineering system these components are needed:

- Documentation
- Tool “ConfigDBCreator”
- Example project Step7 Classic (V5.5)

The Step7 demo project includes all needed function blocks for the user application.

The tool “Config DB Creator” does not need any installation. Just copy the directory to the engineering station. The needed Visual Basic runtime files should exist on a Windows installation (Windows XP or higher).

There is **no** installation needed of the WinAC FileServer driver (setup.bat) on the engineering station.

5.4 Installation WinAC driver on engineering system with Step7 V11 (TIA-Portal)

Platform requirements:

- This documentation
- Step7 V11 (TIA-Portal) demo application
- Tool „WinAC ConfigDB Creator“

The S7 demo application contains the program blocks. You can copy the driver function blocks to your application

For the *„WinAC ConfigDB Creator“* is no installation required.

Only the Visual Basic 6 Runtime Library and a Microsoft Windows XP or higher Operating System is required.

An installation of the WinAC driver (**setup.bat**) is **not** needed on the run-time system.

WinAC - CPU in the project

If the driver function blocks are copied to a project without a WinAC CPU the FBs are marked with an error, because the used SFBs are not supported by other PLCs like WinAC.

Copy the elements (blocks, constants and tables)

To use the WinAC FileServer in a TIA Portal project the following components are needed:

- The driver blocks (folder “Program blocks”)
- Constants (folder “PLC-Variables“ → WinAC_FileServer_Constants),

A second instance of the TIA Portal is needed for copying. Open the target project in the second instance and copy the function blocks (e.g. *“drag and drop”*) from the demo project (WinAC FileServer).

Compiling blocks

In some cases the TIA portal may mark the driver function blocks redbecause some functions of the WinAC ODK (EXEC_COM and CREA_COM) are not recognized. If this happens you have to compile the function blocks again to update the function block’s interfaces.

6 Use Cases of the Application

6.1 Provided Step7 Classic example project

The driver package includes a Step7 Classic (V5.5) example project. The example application is realized in FBD (Function Block Diagram). Target of the example is the demonstration of all functions of the **WinAC FileServer** driver. The different functionalities are controlled by some variable tables.

OB100 Complete Restart

In the beginning the bFirstRun flag is set. It is used for initialization of all FileServer function blocks. Additional internal request flags are reset.

In start up phase the driver is loaded (**FBFileInit**).

OB1 CYCL_EXC

The OB1 contains all the function blocks of the FileServer driver. They can be activated by separate request flags (see variable tables).

At the end the bFirstRun flag is reset.

FB801 – FB806 –FileServer FBs

These are the function blocks of the WinAC FileServer driver.

DB801 – DB806 –FileServer instance data blocks

These are the instance data blocks of the FBs of the WinAC FileServer driver.

DB1000 DBGlob

This data block contains various variables used by this demo project. Thus the example does not need any flags.

DB2000 / 2001, DB 2002 / 2003, DB 2004 / 2005, DB 2006 / 2007, DB 2008 / 2009

These is a collection of data blocks to store plus the matching configuration data blocks.

VAT_... – Variable tables

The variable tables are prepared to test specific functionality of the FileServer driver.

The table VAT_FILE_SERVER controls the functionality of the driver. The remaining tables show content of the various data blocks to store.

6.2 Provided TIA portal example project (V11)

The driver package includes a TIA Poral V11 example project. The example application is realized in FBD (Function Block Diagram). Target of the example is the demonstration of all functions of the **WinAC FileServer** driver. The different functionalities are controlled by some variable tables.

The organization of the TIA portal corresponds to the Step7 classic example. Please refer the documentation of the classic example (see chapter 6.1 “Provided Step7 Classic example project” on page 39).

6.3 Process measure data block with CSV

A data block stores uniform data (e.g. measure data: timestamp, value 1, value 2, etc.). This data block should be written in a CSV file or read from CSV file.

The data block may have the following structure:

Table 6-1 Example data block with measure values

```
STRUCT
  data : ARRAY[0.999] OF
    STRUCT
      timestamp : TIME_OF_DAY ;
      voltage : REAL ;
      current : REAL ;
      current : REAL ;
    END_STRUCT ;
END_STRUCT ;
```

To fulfill the requirement there are different approaches:

- Write whole data block with one function call “as a whole”
- Write data block line by line
- Write whole data block with the repeat counter

Write whole data block with one function call “as a whole”

One can create a Config-DB with the structural information for the whole measure value data block (tool ConfigDBCreator).

With one call of FBFileOpen and FBFileWrite the data block can be written.

Drawback: The configuration data block will get big size. It may contain more items the WinAC FileServer supports.

Write data block line by line

In this case the Config-DB describes the measure data, only.

After opening with FBFileOpen with APPEND flag set, the FBFileWrite must be called many times – one time for every measure value. The input parameter DATA_DB (ANY) must point to the next measure value.

Drawback: Because of the many single calls of FBFileWrite the writing will cost some time.

Tip To create the Config-DB for the measure data only with the tool ConfigDBCreator one can reduce the size of the array to 1 temporaly.

Write whole data block with the repeat counter

Also in this case the Config-DB describes the measure data, only.

Now the FBFileWrite is called with REPEAT_CNT = 1000. Thus this single call processes the whole data block containing many measure values.

6.4 Access network drives

The WinAC including their extensions like “WinAC FileServer” runs as user SYSTEM, i.e. system process. A system process runs with “Zero Credentials”, i.e. it is allowed only to access local resources.

By default a WinAC driver cannot access any network drives. But it is possible to create a so called “null session share” which can be assessed without login name and password.

Note The following settings have to be done on the remote PC. On this PC the directory is located for storing WinAC data.

- Create directory and share
  e.g. “WinACData”; full access for user “Everyone”
- Group Policy Editor (start e.g. by gpedit.msc)
- Computer configuration \ Windows settings \ Security Settings \ Local settings \ safety options
  + “Network access: shares that can be accessed anonymously”:
    add “WinACData” here
  + “Network access: Let everyone permissions apply to anonymous users”
    activate this

Note The path in the Step7 project for FBFileOpen must be provided in UNC notation.

7 Error Codes

The WinAC FileServer can provide different classes of error messages:

- Code in the FB-output **STATUS** according to WinAC-ODK (see chapter 8.1 in this document)
- Special error codes of the FileServer (see chapter 8.2 on page 45 in this document)

7.1 Error codes of WinAC ODK

The driver had been developed with the WinAC ODK (Open Development Kit). The ODK can generate error codes, which are returned from the **STATUS** of the FBs.

Table 7-1 WinAC ODK error messages

ODK Code (HEX)	Description
0	Success
8001	An exception occurred.
8002	Input: the ANY pointer is invalid.
8003	Input: the ANY pointer range is invalid.
8004	Output: the ANY pointer is invalid.
8005	Output: the ANY pointer range is invalid.
8006	More bytes were written into the output buffer by the extension object than were allocated.
8007	ODK system has not been initialized: no previous call to SFB65001 (CREA_COM).
8008	The supplied handle value does not correspond to a valid extension object.
8009	More bytes were written into the input buffer by the extension object than were allocated.
807F	An internal error occurred.
80C3	Maximum number (32) of parallel jobs/instances exceeded.
8102	The call to CLSIDFromProgID failed.
8103	The call to CoInitializeEx failed.
8104	The call to CoCreateInstance failed.
8105	The library failed to load.
8106	A Windows response timeout occurred.
8107	Controller is in an invalid state for scheduling an OB.
8108	Schedule information for OB is invalid.
8109	Instance ID for SFB65001 call is invalid.
810A	Controller could not load proxy DLL.
810B	The WinAC controller could not create or initialize shared memory area.
810C	Attempt to access unavailable option ocurred.
8201	The Execute command index could not be found
8250	No more available positions in the job list

ODK Code (HEX)	Description
8252	The count is invalid
8253	A data type of an item in the list is invalid
8254	The count specified is invalid
8255	A memory area of an item in the list is invalid
8256	A DB number of an item in the list is invalid
8257	A bit number of an item in the list is invalid
8258	A pBuff of an item in the list is invalid
8259	A data quantity is invalid
825A	The area offset parameter is invalid for this type
825B	The frequency value is invalid
825C	The callback pointer is invalid
825D	The job ID pointer is invalid
825E	The job ID is invalid
825F	Job could not be completed because address is incorrect
8260	Job could not be completed because of protection level
8261	Job could not be completed because of hardware issues
8301	Invalid Thread Execution Priority
8401	Invalid Asynchronous Event
8402	Asynchronous Processor Queue is empty
8403	Asynchronous Processor Queue is full

7.2 Special error codes of the WinAC File Server

Among the general error bit of the driver FBs there is a special error code in the value of **STATUS** to describe the reason of the problem.

Note Several error reasons provide additional information in the instance DB of the function block (tOdkIf.dwErrInfo1 … 4)

Table 7-2 Error codes of WinAC FileServer

0 - no error **Interface to WinAC** 0x8501 - error using ODK_Read.. function 0x8502 - error using ODK_Write.. function 0x8503 - no config DB given 0x8504 - no config DB given 0x8505 - no read DB given 0x8506 - no read DB given 0x8507 - no read DB given 0x8508 - no read DB given 0x850A - no write DB given 0x850B - no write DB given 0x8510 - false version of Step7 function block 0x8511 - false ID for file 0x8512 - not supported file format **State errors** 0x8520 - no successful init called before 0x8521 - update config is active 0x8522 - read file is active, update config not allowed 0x8523 - write file is active, update config not allowed 0x8524 - unknown internal state (no 'file open' called before?) 0x8525 - false internal state after changing config 0x8526 - false internal state after reading file 0x8527 - false internal state after writing file **Analysing the configuration** 0x8531 - no file defined yet 0x8532 - config DB version not supported 0x8533 - to many items for internal storage 0x8534 - no start index of array found 0x8535 - no start index of array found 0x8536 - append mode not allowed for BIN files 0x8537 - append mode not allowed for INI files 0x8538 - no strlen for string found 0x8539 - max. strlen to small (0 or negative) 0x853a - max. strlen to long (> 254) 0x853b - unsupported data type found 0x853c - internal error - exception catched

Errors writing file
0x8540 - no valid repeat counter for writing
0x8541 - no file handler (file format) assigned to this file ID
0x8542 - error open file for writing
0x8543 - error writing to file
0x8544 - internal error - exception catched

0x8551 - cannot write data outside of struct (needed for section name)
0x8552 - nested structs not allowed for INI files
0x8553 - error writing parameter to INI file

0x8561 - unexpected data type when writing XML writing basic type
0x8562 - unexpected data type when writing XML writing array
0x8563 - unexpected data type when writing XML writing struct

0x8564 - error seeking file from end
0x8565 - error file get pos
0x8565 - error file fgetc

0x8571 - error writing BIN file

0x8581 - this S7 data type is not supported for writing
0x8582 - error writing BOOL - reading from WinAC
0x8583 - error writing BYTE - reading from WinAC
0x8584 - error writing CHAR - reading from WinAC
0x8585 - error writing CHAR - range of character
0x8586 - error writing WORD - reading from WinAC
0x8587 - error writing DWORD - reading from WinAC
0x8588 - error writing INT - reading from WinAC
0x8589 - error writing DINT - reading from WinAC
0x858a - error writing REAL - reading from WinAC
0x858b - error writing S5TIME - reading from WinAC
0x858c - error writing S5TIME - problem with time base
0x858d - error writing STRING - internal buffer to small
0x858e - error writing STRING - reading from WinAC
0x858f - error writing TIME - reading from WinAC
0x8590 - error writing TIME_OF_DAY - reading from WinAC
0x8591 - error writing DATE - reading from WinAC
0x8592 - error writing DATE - range error
0x8593 - error writing DATE_AND_TIME - reading from WinAC

Error reading a file
0x8600 - no valid repeat counter for reading
0x8601 - error open file for reading
0x8602 - internal error - exception catched

0x8611 - error getting file status
0x8612 - error file size does not match DB size
0x8613 - error reading file binary

0x8621 - no header line found in CSV file
0x8622 - end of line before all values read
0x8623 - end of file before all values read
0x8624 - value string too long
0x8625 - value string too short (CSV file to short?)

0x8631 - given INI parameter/section not found

0x8641 - a line in ASCII file is too long
0x8642 - line without '='
0x8643 - unknown item found in ASCII file
0x8644 - got invalid index for item

0x8651 - unknown item found in XML file
0x8652 - got invalid index for item
0x8653 - empty parent

0x8700 - base address for XML read errors
0x8701 - XML read - Error
0x8702 - XML read - Failed to open file
0x8703 - XML read - Memory allocation failed.
0x8704 - XML read - Error parsing Element.
0x8705 - XML read - Failed to read Element name
0x8706 - XML read - Error reading Element value.
0x8707 - XML read - Error reading Attributes.
0x8708 - XML read - Error empty tag.
0x8709 - XML read - Error reading end tag.
0x870A - XML read - Error parsing Unknown.
0x870B - XML read - Error parsing Comment.
0x870C - XML read - Error parsing Declaration.
0x870D - XML read - Error document empty.
0x870E - XML read - Error null (0) or unexpected EOF found in input stream.
0x870F - XML read - Error parsing CDATA.
0x8710 - XML read - Error when XmlDocument added to document,
because XmlDocument can only be at the root.
0x8711 - XML read - Error

Error reading file - parsing value strings
0x8801 - this type is not supported for reading
0x8802 - error reading data type 'BOOL' - writing 'true' to WinAC
0x8803 - error reading data type 'BOOL' - writing 'false' to WinAC
0x8804 - error reading data type 'BOOL' - unknown value
0x8805 - error reading data type 'BYTE' - writing to WinAC
0x8806 - error reading data type 'CHAR' - got no chacter
0x8807 - error reading data type 'CHAR' - got string with lenght > 1
0x8808 - error reading data type 'CHAR' - out of range
0x8809 - error reading data type 'CHAR' - writing to WinAC
0x880a - error reading data type 'WORD' - writing to WinAC
0x880b - error reading data type 'DWORD' - writing to WinAC
0x880c - error reading data type 'INT' - writing to WinAC
0x880d - error reading data type 'INT' - writing to WinAC
0x880e - error reading data type 'REAL' - writing to WinAC
0x880f - error reading data type 'STRING'
0x8810 - error converting hex value (false prefix?)

0x8812 - error reading data type 'BYTE' - range error
0x8814 - error reading data type 'WORD' - range error
0x8816 - error reading data type 'INT' - range error: value to big
0x8817 - error reading data type 'INT' - range error: value to small

0x8821 - error reading S5TIME - writing value to WinAC
0x8822 - error reading S5TIME - false prefix ('S5T#')
0x8823 - error reading S5TIME - [MS] resolution (divisible by 10)
0x8824 - error reading S5TIME - [MS] value

0x8825 - error reading S5TIME - [S] value
0x8826 - error reading S5TIME - [MIN] value
0x8827 - error reading S5TIME - [H] value
0x8828 - error reading S5TIME - value out of range
0x8829 - error reading S5TIME - value out of range
0x882a - error reading S5TIME - base out of range

0x8831 - error reading STRING - invalid pointer to string
0x8832 - error reading STRING - error reading string len
0x8833 - error reading STRING - string is too large for the STEP 7 string
0x8834 - error reading STRING - string is too large for the output data buffer
0x8835 - error reading STRING - error writing current string len
0x8836 - error reading STRING - error writing max. string len
0x8837 - error reading STRING - error writing string to WinAC

0x8841 - error reading TIME - prefix 'T#' is missing
0x8842 - error reading TIME - [MS] value
0x8843 - error reading TIME - [S] value
0x8844 - error reading TIME - [MIN] value
0x8845 - error reading TIME - [H] value
0x8846 - error reading TIME - [D] value
0x8847 - error reading TIME - range exceed
0x8848 - error reading TIME - writing value to WinAC

0x8851 - error reading TIME_OF_DAY - prefix 'TOD#' is missing
0x8852 - error reading TIME_OF_DAY - writing value to WinAC

0x8861 - error reading DATE - prefix 'D#' is missing
0x8862 - error reading DATE - range year exceed
0x8863 - error reading DATE - range month exceed
0x8864 - error reading DATE - range day exceed
0x8865 - error reading DATE - range exceed: computing seconds
0x8866 - error reading DATE - writing value to WinAC

0x8871 - error reading DATE_AND_TIME - prefix 'DT#' is missing
0x8872 - error reading DATE_AND_TIME - writing value to WinAC
0x8873 - error reading DATE_AND_TIME - [MS] value
0x8874 - error reading DATE_AND_TIME - [S] value
0x8875 - error reading DATE_AND_TIME - [MIN] value
0x8876 - error reading DATE_AND_TIME - [H] value
0x8877 - error reading DATE_AND_TIME - [DAY] value
0x8878 - error reading DATE_AND_TIME - [MON] value
0x8879 - error reading DATE_AND_TIME - [Y] value
0x887A - error reading DATE_AND_TIME - computing weekday

Other errors
0x8901 - write not implemented in reader classes
0x8902 - undefined return value

Errors file handling
0x8A01 - error deleting file
0x8A02 - error copying file
0x8A03 - error moving file
0x8A04 - internal error - exception catched

0x8A11 - error retrieving free disk space

0x8A12 - error retrieving free disk space
0x8A13 - error creating file
0x8A14 - error get file size
0x8A15 - error get file time
0x8A16 - error converting file time
0x8A17 - error get file attributes
0x8A18 - internal error - exception catched

Errors generated in STEP7 FBs
0x9D01 - false file ID given
0x9D02 - no command selected (no command flag set)
0x9D03 - To much commands. only one comand on request is allowed.

8 Related Literature

8.1 Bibliography

This list is not complete and only represents a selection of relevant literature.

Table 8-1

	Subject	Title
/1/	STEP7	Automation with STEP7 in STL and SCL Hans Berger Publisher: Vch Pub ISBN-10 3895783412 ISBN-13 9783895783418
/2/	WinAC	Windows Automation Center RTX – WinAC RTX 2010 Operating Instructions
/3/		

8.2 Internet Link Specifications

This list is not complete and only represents a selection of relevant information.

Table 8-2

	Subject	Title
\1\	Reference to the entry	http://support.automation.siemens.com/WW/view/en/EntryID
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\3\	WinAC Operating Instructions	http://support.automation.siemens.com/WW/view/en/43715176
\4\		

9 History

Table 9-1 Version History

Version	Date	Modifications
V1.0.0	02.02.11	First version
V 1.1.0	14.03.11	- File handling FBs added to documentation - New function 'repeat counter' for reading / writing CSV files
V1.2.0	26.09.11	- New Layout for standard applications - Description of the Step7 example project - new output "DONE" for all function blocks - Reading Bool value accepts "0"/"1", too - Description for installation under Windows XP and Windows 7 - Time charts for REQ, BUSY, DONE and ERROR - Additional information to INI file format - Note: data type ANY and POINTER not supported - Note: XML file format is fixed - RepeatCounter in FB description and use case chapter - XML example "array of struct"
V1.2.3	01.03.12	- TIA Portal V11 example added - Extension of tool "Config DB Creator" for incorporation with TIA portal - Increase of internal quantities (2.000 items → 4.000)