How can you output batch data or archived data via an HMI operator panel?

WinCC flexible 2008 SP2

FAQ • September 2011

Service & Support

Answers for industry.

SIEMENS

Note on security
The functions and solutions described in this article confine themselves predominantly to the realization of the automation task. Furthermore, please 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. Further information can be found in Entry ID: !50203404!.

Question

How can you output batch data or archived data via an HMI operator panel?

Answer

Follow the instructions and notes listed in this document for a detailed answer to the above question.

Contents

1 Introduction

Aim of the entry

The aim of this entry is to show a way of how to archive batch data or do long-term archiving with WinCC flexible.

In addition, the entry shows you how to export and save the archived entries in a "*.csv file".

In order also to be able to use operator panels that do not support scripts, we have not used scripts in this sample application.

Which operator panels are supported?

The instructions below apply for all operator panels that support the following functions:

- Recipes
- Export data records
- Slot for a memory card

An overview of the functions of the various operator panels is available in Entry ID 40227286.

In this sample application we have used a TP177B PNDP 4".

Brief description of the application

The application includes two sample applications:

- **Example 1:**
  The first example is simple and is designed to show the function
  - Reading out of values from a data block using two "recipe functions" of the HMI operator panel and the subsequent exporting of the data as "*.csv file".
  The data of the data block is fixed in this example and represents the "machine parameters" that are output as batch data. As user, you would provide this data for the program in the
  S7 controller and assign it to the relevant data block.
- **Example 2:**
  In Example 2 the same functionality as in Example 1 is used.
  The difference is that with "Start of the machine" the measured values are read out every second and stored consecutively in a data block.
  At the beginning of each minute these values are read out automatically and stored in a "*.csv file".
  The name of the "„*.csv file" is created automatically via the STEP 7 program and transferred as file name.

  The name of the file is composed as follows:
  \storage path\name_date_time.csv

Example:
\Storage Card MMC\LogData_110810_1419.csv

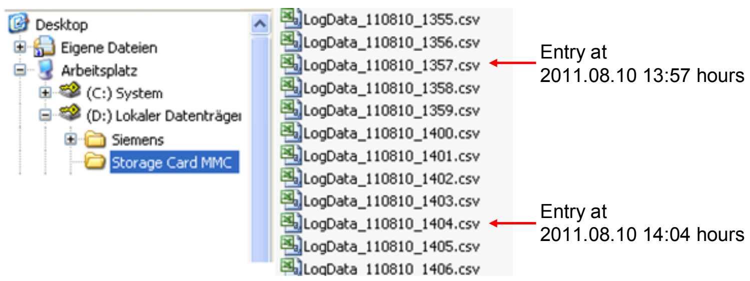
- Storage path: **\Storage Card MMC**
- Name: **LogData**
- Date: **11** (year) **08** (month) **10** (day)
- Time of day: **14** (hour) **19** (minute)

Notes

The storage path, name and format of the date and time can be changed as required via the STEP 7 program.

Example of archived data

Figure 1-1



Testing the application

The attached configuration can also be tested with S7-PLCSIM and the WinCC flexible Runtime.
(S7-PLCSIM and the Runtime software must be installed on the configuration computer).

2 Automation Solution

Background information for the automation solution

Siemens offers a wide range of HMI operator panels with different features.
Not all HMI operator panels support the functions below, for example:

- Tag archives
- Scripting

Notes

Furthermore, the system limits for archiving tag values on operator panels and PC Runtime systems permit only restricted long-term archiving. In the case of an MP 277, the limit is 10,000 entries per archive (including all the archive segments).

Example
If you log a tag every second, then the system limit of 10,000 entries is reached after about 2.5 hours. If you were to use a "circular archive", the oldest entries would be overwritten with the new values.

The FAQ response in Entry ID 48015332 gives a sample application for long-term archiving using scripts.

This example **does not use any scripts.** In this way, HMI operator panels with lower performance can also be used.

2.1 How Does Configuration Work?

In this case, creating/archiving data is **not** done via "Tag archives" on the HMI operator panel, but via the data blocks in the
S7 controller.

The current value of a machine parameter is queried via a "Trigger" (pulse) every second, for example, and then stored in a data block. Each separate value is assigned automatically to a data word in the data block.

The output of this "archived" data from the data block is done on the
HMI operator panel via a configured **recipe**.

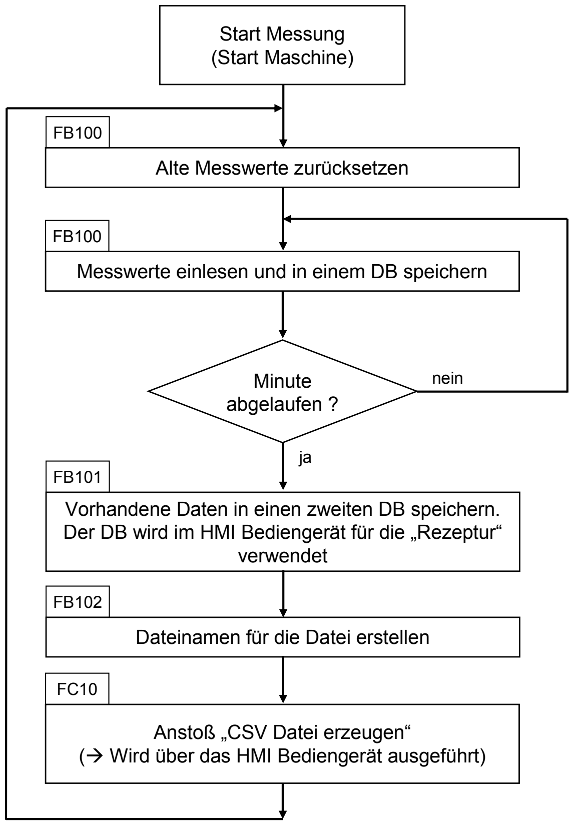
With recipes there is the option of reading out the values from the controller and then export them as a "*.csv" file to a memory card.

In this way you can archive the data in the S7 controller as a "**file**" on a memory card.

2.2 Overview

The figure below gives an overview of the separate functional sequences for Example 2.

Figure 2-1


Start Messung
(Start Maschine)
FB100
Alte Messwerte zurücksetzen
FB100
Messwerte einlesen und in einem DB speichern
Minute abgelaufen ?
nein
ja
FB101
Vorhandene Daten in einen zweiten DB speichern. Der DB wird im HMI Bediengerät für die „Rezeptur“ verwendet
FB102
Dateinamen für die Datei erstellen
FC10
Anstoß „CSV Datei erzeugen“
(→ Wird über das HMI Bediengerät ausgeführt)

Description

When the measurement starts or at the beginning of a new "minute" the existing data is first reset to "zero".

Then a measured value is read out every second and stored in a data block.

The archiving is done over a period of one minute.

When the minute expires, the next step is executed via the STEP 7 program.

- The data archived before is copied to a second data block. The second data block is used in a "recipe" in the HMI operator panel.
- A file name is created for the "*.csv file" to be exported.
- The functions for reading out the data from the controller and then exporting the data are triggered via the HMI operator panel.

3 Configuration

This chapter describes the configuration steps to be taken to readjust the configuration.

- HMI configuration
- STEP 7 program

Details are to be found in the attached project.
For better understanding it is useful to open the attached configuration.

Notes All the settings already described have been made in the attached configuration.

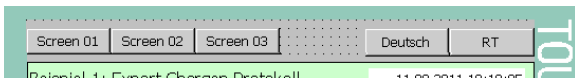
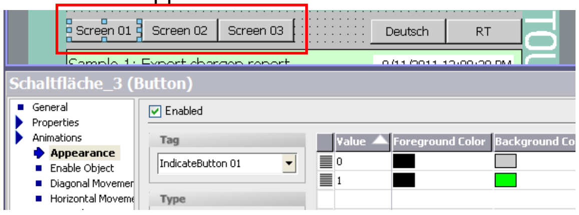
3.1 HMI Configuration

3.1.1 Configured Screens

Three screens are configured in the attached configuration. The functions configured in the screens are described below.

Permanent window

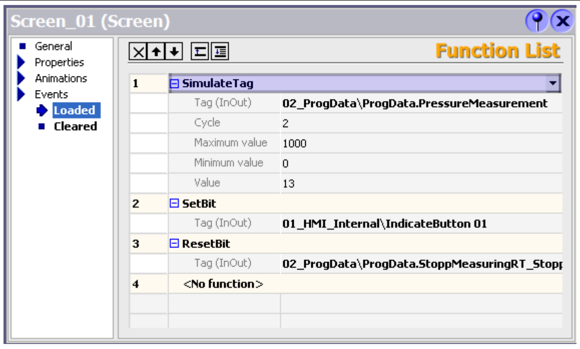
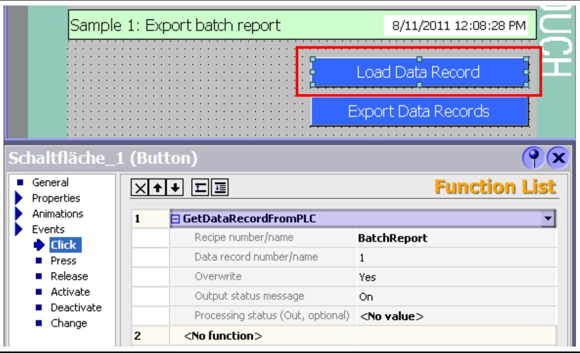
Table 3-1

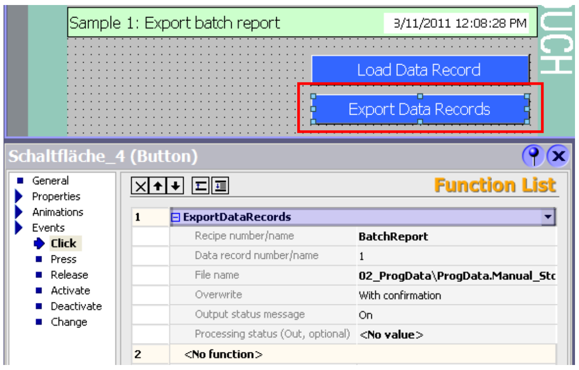
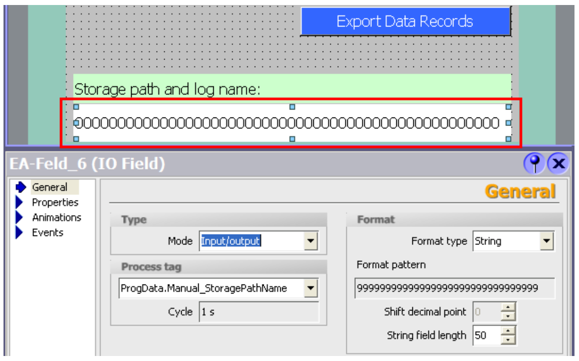
No.	Description	Screen
1.	**Permanent window** Four buttons are configured in the permanent window. These buttons are for ... • Calling individual screens directly. • Changing the language of the user interface (German/English). • Ending Runtime. An animation is configured for each of the buttons "Screen 1", "Screen 2" and "Screen 3" (Properties > Animation > Appearance). When the corresponding screen is opened the background color of the button is "green". The tag is set when the screen opens and reset when the screen closes again (Link). In this way you can see in which screen you are located.	Permanent window:  == Animation – Appearance:  Screen 01 Screen 02 Screen 03 Deutsch RT General Properties Animations Appearance Enable Object Diagonal Movemen Horizontal Moveme Schaltfläche_3 (Button) ☑ Enabled Tag: IndicateButton 01 Type Value | Foreground Color | Background Co 0 | ■ | □ 1 | ■ | 🟩

Screen 1

Example 1 is operated with "Screen 1".

Table 3-2

No.	Description	Pictures
1.	**„"Screen 1" – Screen properties** Open the properties of "Screen 1". • **Events - Loaded** - „SimulateTag“ The "measured values" for "Example 2" are created with the "SimulateTag" function. „SetBit“ The background color of the "Screen 1" button is changed with an animation. When the screen is loaded, the "bit" of the assigned tag is set (Link). "ResetBit" When Runtime ends, a "bit" is reset via the "scheduler". When you restart the operator panel, this bit is reset by this "home page". • **Events - Loaded** - „"ResetBit" The background color of the "Screen 1" button is changed by means of an animation. When the screen is cleared, the "bit" of the assigned tag is reset.	 Screen_01 (Screen) General / Properties / Animations / Events: **Loaded**, Cleared Function List 1 SimulateTag Tag (InOut): 02_ProgData\ProgData.PressureMeasurement Cycle: 2 Maximum value: 1000 Minimum value: 0 Value: 13 2 SetBit Tag (InOut): 01_HMI_Internal\IndicateButton 01 3 ResetBit Tag (InOut): 02_ProgData\ProgData.StoppMeasuringRT_Stopp 4 <No function>
2.	**„"Screen 1" – "Load Data Record"** With the button the data record is read out of the controller. The data record contains the batch data for the stored machine values (Link).	 Sample 1: Export batch report 8/11/2011 12:08:28 PM Schaltfläche_1 (Button) General / Properties / Animations / Events: **Click**, Press, Release, Activate, Deactivate, Change Function List 1 GetDataRecordFromPLC Recipe number/name: BatchReport Data record number/name: 1 Overwrite: Yes Output status message: On Processing status (Out, optional): <No value> 2 <No function>

No.	Description	Pictures
3.	**„"Screen 1" – "Export Data Record"** With this button you export the previously loaded data record as a "*.csv file" and store it on a memory card. The name of the "*.csv file" can be specified as required via the "text field" described below.	
4.	**„"Screen 1" – "Text field"** You use the text field to specify a file name as required for the "*.csv file" to be exported. The tag is preset with a "default name" (storage path + file name + .csv). **Example:** \Storage Card MMC\DataLog_xxx.csv In this way, the operator only has to change/enter the "file name".	

Sample 1: Export batch report 3/11/2011 12:08:28 PM

Schaltfläche_4 (Button) – Function List – Events: Click

1	ExportDataRecords	
	Recipe number/name	BatchReport
	Data record number/name	1
	File name	02_ProgData\ProgData.Manual_Stc
	Overwrite	With confirmation
	Output status message	On
	Processing status (Out, optional)	<No value>
2	<No function>	

Storage path and log name:

EA-Feld_6 (IO Field) – General

Type – Mode: Input/output

Process tag: ProgData.Manual_StoragePathName

Cycle: 1 s

Format – Format type: String

Format pattern: 99999999999999999999999999999999

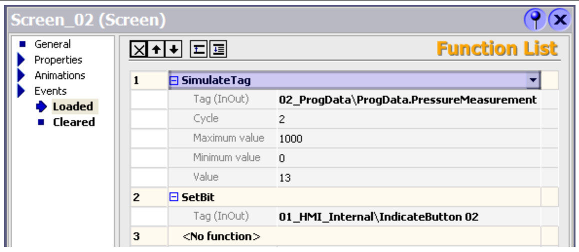
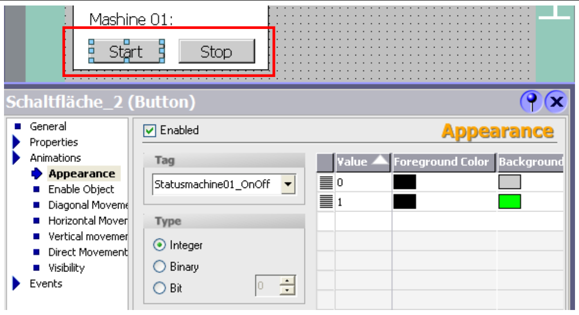
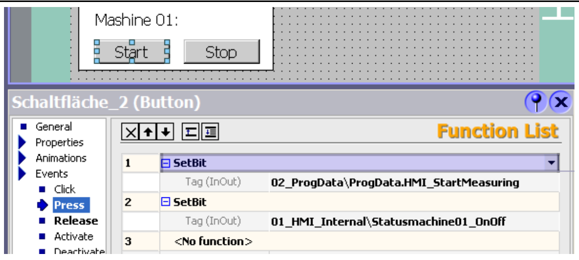
Shift decimal point: 0

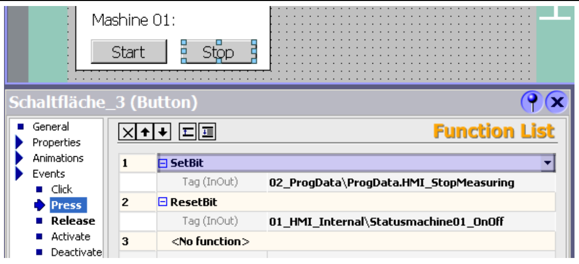
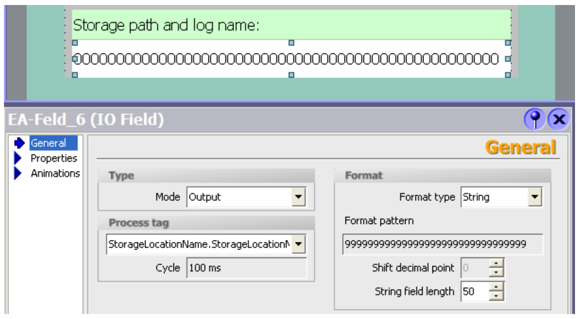
String field length: 50

Screen 2

Example 2 is operated with "Screen 2".

Table 3-3

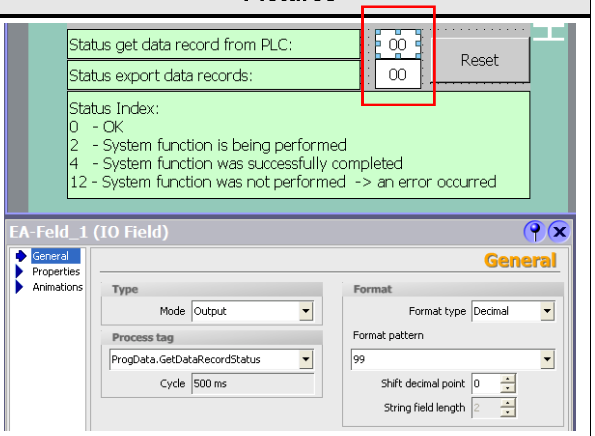
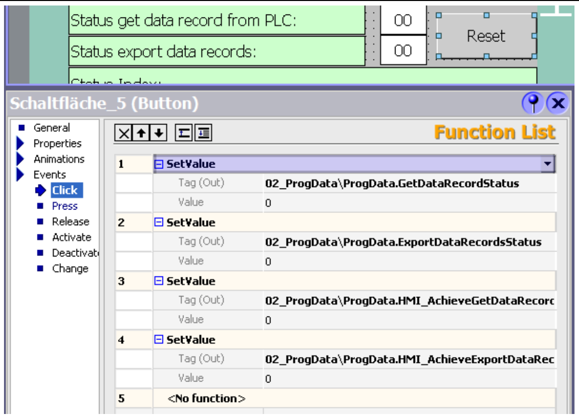
No.	Description	Pictures
1.	**„"Screen 2" – Screen properties** Open the properties of "Screen 2". • **Events - Loaded** - „"SimulateTag" The "measured values" for the "Example 2" are generated with the "SimulateTag" function. "SetBit" The background color of the "Screen 2" button is changed by means of an animation. When the screen is loaded, the "bit" of the assigned tag is set (Link). • **Events - Loaded** - „"ResetBit" The background color of the "Screen 2" button is changed by means of an animation. When the screen is cleared, the "bit" of the assigned tag is reset.	 Screen_02 (Screen) General Properties Animations Events Loaded Cleared Function List 1 SimulateTag Tag (InOut) 02_ProgData\ProgData.PressureMeasurement Cycle 2 Maximum value 1000 Minimum value 0 Value 13 2 SetBit Tag (InOut) 01_HMI_Internal\IndicateButton 02 3 <No function>
2.	**„"Screen 2" – Property buttons** **Animation:** An animation is configured for each of the buttons "Start" and "Stop". Properties > Animation > Appearance. When you click the relevant button, the background color of the button is "green" or "orange".	 Mashine 01: Start Stop Schaltfläche_2 (Button) General Properties Animations Appearance Enable Object Diagonal Moveme Horizontal Mover Vertical movemer Direct Movement Visibility Events Appearance ☑ Enabled Tag: Statusmachine01_OnOff Type: [✓ Integer] Binary Bit 0 Value / Foreground Color / Background 0 ■ ⬜ 1 ■ 🟩
3.	**„"Screen 2" – "Start" property button** **Events – Press:** When you press the button the bit to start "measurement" is set in the controller. (In this example we do not switch on a "real" drive). Furthermore, the bit for "animation" of the button is set. **Events - Release:** When you release the button the bit to start "measurement" is reset.	 Mashine 01: Start Stop Schaltfläche_2 (Button) General Properties Animations Events Click Press Release Activate Deactivate Function List 1 SetBit Tag (InOut) 02_ProgData\ProgData.HMI_StartMeasuring 2 SetBit Tag (InOut) 01_HMI_Internal\Statusmachine01_OnOff 3 <No function>

No.	Description	Pictures
4.	**„"Screen 2" – "Stop" property button** **Events – Press:** When you press the button the bit to stop "measurement" is set in the controller. (In this example we do not switch off a "real" drive). Furthermore, the bit for "animation" of the button is reset. **Events - Release:** When you release the button, the bit to start "measurement" is reset.	 Mashine 01: Start Stop Schaltfläche_3 (Button) General / Properties / Animations / Events: Click, **Press**, Release, Activate, Deactivate Function List 1 SetBit Tag (InOut) 02_ProgData\ProgData.HMI_StopMeasuring 2 ResetBit Tag (InOut) 01_HMI_Internal\Statusmachine01_OnOff 3 <No function>
5.	**„"Screen 2" – "Text field"** The storage path of the "*.csv file" to be exported is output in the text field. The name of the file is set automatically by the STEP 7 program. (storage path + file name + date + time.csv) **Example:** \Storage Card MMC\DataLog_110812_1637.csv	 Storage path and log name: 0000000000000000000000000000000000000000000000000 EA-Feld_6 (IO Field) General / Properties / Animations General Type: Mode Output Process tag: StorageLocationName.StorageLocationN Cycle 100 ms Format: Format type String Format pattern 99999999999999999999999999999999 Shift decimal point 0 String field length 50

Screen 3

"Screen 3" is the “service site" for "**Example 2**". The statuses of the separate "recipe functions" are output here. If there is an error, you can also reset the status here.

Table 3-4

No.	Description	Pictures
1.	**„"Screen 3" – "IO fields"** The statuses of the recipe functions below are output via the IO fields • Get data record from PLC • Export data records The possible statuses and their meanings are listed in the lower area of the screen.	 Status get data record from PLC: 00 Status export data records: 00 Status Index: 0 - OK 2 - System function is being performed 4 - System function was successfully completed 12 - System function was not performed -> an error occurred EA-Feld_1 (IO Field) General Type: Mode: Output Process tag: ProgData.GetDataRecordStatus Cycle: 500 ms Format: Format type: Decimal Format pattern: 99 Shift decimal point: 0 String field length: 2
2.	**„"Screen 3" – "Reset" property button** **Events - Click:** When you press this button, the value "**0**" is transferred to the four tags. **Note:** By default the values are reset to "0" automatically by the STEP 7 program. If there is an error and this does not happen, you can do a reset manually using the "Reset" button. (For this see also the description at this link).	 Status get data record from PLC: 00 Status export data records: 00 Schaltfläche_5 (Button) Events – Click – Function List 1 SetValue Tag (Out): 02_ProgData\ProgData.GetDataRecordStatus Value: 0 2 SetValue Tag (Out): 02_ProgData\ProgData.ExportDataRecordsStatus Value: 0 3 SetValue Tag (Out): 02_ProgData\ProgData.HMI_AchieveGetDataRecorc Value: 0 4 SetValue Tag (Out): 02_ProgData\ProgData.HMI_AchieveExportDataRec Value: 0 5 <No function>

3.1.2 Tags

The tags are grouped in "subfolders" according to their use.

Some tags have their "properties" changed with reference to the default setting like "Acquisition type", for example.

These tags are dealt with below.

Tag folder "01_HMI_Internal"

This folder contains tags that do not have a controller connection. They serve on buttons for "graphical" signalization of "switching statuses" (Link).

Tag folder "02_ProgData"

The tags all have a controller connection. They serve to execute "program functions" like "Start the measurement", for example.

Table 3-5

No.	Tag	Description
1.	StorageLocationName **General > Data type:** StringChar	The file name of the "*.csv file" is preset via the STEP 7 program for "Example 2". By using "StringChar" you can transfer texts without "string header". **"String header":** The "string header" contains information about the maximum and current lengths of the string.
2.	StopMeasuringRT_Stop **General > Acquisition type:** Cyclic continuous	This tag is set by the scheduler when WinCC flexible Runtime ends. This interrupts a running measurement in the controller. **Note:** Before ending Runtime, the operator should end a running measurement (switch off machine). If you forget to do this, the running measurement is terminated when Runtime ends.
3.	HMI_AchieveGetDataRecord **General > Acquisition type:** Cyclic continuous **Properties > Limits:** Upper limit: 2 **Events > Upper limit exceeded** System function	The reading out of values from the controller to the HMI operator panel is done automatically in "Example 2". In order for this to be done, the controller presets a value > 2 and thus the specified upper limit is exceeded. The "Get data record from PLC" system function is executed.

No.	Tag	Description
4.	HMI_AchieveExportDataRec **General > Acquisition type:** Cyclic continuous **Properties > Limits:** Upper limit: 2 **Events > Upper limit exceeded** System function	Export of the values read out to an "MMC memory card" is done automatically in "Example 2". In order for this to be done, the controller presets a value > 2 and thus the specified upper limit is exceeded. The "Export data records" system function is executed.

Tag folder "03_BatchProtocol"

The tags are all assigned to the "**BatchReport**" recipe and contain the data acquired from the control program (in this case batch data).

You can save/export these tags as a "*.csv file" to a memory card.

Tag folder "04_LogData"

The tags are all assigned to the "**LogData**" recipe and contain the data archived from the control program for "Machine 01".

You can save/export these tags as a "*.csv file" to a memory card.

3.1.3 Recipes

The configured recipes are for reading the archived data out of the controller program and saving/exporting it to a memory card.

The configuration therefore does not contain a "recipe display".
(Can also be configured in addition in the HMI operator panel).

The number of tags that can be used depends on the quantity framework of the operator panel used.

"BatchReport" recipe

The recipe contains the "batch data" collected from data block DB15. The recipe contains a data record.

"LogData" recipe

The recipe contains the archived machine data of Machine 01.
The recipe has a total of 60 entries - one value for each second.
The values were stored previously in DB12 and then copied to DB13. The recipe contains a data record.

3.1.4 Other Functions Used

Scheduler

The scheduler sets a bit in the controller when Runtime ends.

Background:
The STEP 7 program triggers the "Get data record from PLC" and "Export data records" recipe functions.
The controller program evaluates the processing status of each "recipe function". But this can only happen when the Runtime of the HMI operator panel is online. This is why it is important to terminate a running measurement before ending Runtime.

The programmer has to decide how this is to be monitored.

The scheduler does the job in this application.

3.2 STEP 7 Configuration

The configured FBs have input and output parameters so that little change is needed in the FBs to adapt them to your individual requirements.
The FBs and FCs have the corresponding headers and comments.

Below are explanations of the program blocks used.

OB1, CYCL_EXC

The separate program blocks below are called via OB1:

- FB100, PLC_LogData
- FB101, PLC_CopyLogData
- FB102, PLC_StoragePath
- FC10, HMI_CopyLogData

OB35, CYC_INT5

OB35 is used to generate a "second-trigger" for application example 2.

OB35 is called every 500 ms for this.

FB100, PLC_LogData

FB100 reads in the specified "measured values" and stores them in a data block.

Addressing (assignment of measured values in the DB) is done via a pointer. The current second of the PLC time is evaluated as "index" for the pointer.

The values are saved every second (OB35 trigger).
Every measured value is assigned to a data word.

FB101, PLC_CopyLogData

FB101 executes two functions.

- Determination of when a new minute begins.
  - SFC1 reads out the current time of the PLC.
    The current minute is determined from this time.
    A bit is set in the controller at the beginning of a new and the signal is processed.
- Data copied (DB12 -> DB13).
  - The sample application is configured so that at the beginning of a new minute recipe functions save/export
    the previously stored measured values as *.csv file to a memory card via the HMI operator panel.
    In this case the data of DB12 is copied to DB13.
    DB13 is used for the "**LogData**" recipe in the HMI operator panel.

FB102, PLC_StoragePath

FB102 prepares the storage path and file name of the "*.csv file" for "Runtime".

The name is composed of the elements below:

- Storage path + file name + date + time + .csv

  Example:
  \Storage Card MMC\ LogData_110815_1458.csv

The storage path and the file name are firmly specified here.

- \Storage Card MMC\LogData_ (see DB110)

At "Runtime", the current date and time are added to the preset name.

In this way each file receives a unique name.

When configuring you should note that "one" minute must be subtracted from the current time.

Background:
The file is saved **after** the end of one full minute. This means that the archived values are not from the "current" minute, but from the "previous" minute. For this reason the current time is not used for the file name.
Exception: The current time is used when a running measurement is terminated.

FC10, HMI_CopyLogData

FC10 is for enabling the "Get data record from PLC" and "Export data records" functions via the S7 controller.

Example:
The "ProgData.HMI_AchieveGetDataRecord" tag is used in the HMI operator panel. The "Get data record from PLC" system function is executed on this tag when a specified upper limit is exceeded.

The controller program assigns this tag a value > 2 which means that the system function is executed on the HMI operator panel.

The status feedback of the system function is evaluated to determine when the function in completed. Then the second system function, "Export data records", is triggered.

Notes The status feedback is generated automatically by the HMI system function used and read out accordingly. For this reason it is important to terminate a running measurement before ending the HMI Runtime.

DB10, ProgData

DB10 contains general data for implementing the sample applications.

DB12, Current_LogData

DB12 contains the currently stored machine values of "Machine 01".

The data block is configured to store 60 values (1 minute).

DB13, Copy_LogData

DB13 contains the data copied from DB12 and is used for the "recipe" in the HMI operator panel.

The data block is configured to store 60 values (1 minute).

DB14, Overwrite_LogData

Once the data is copied from DB12 to DB13, the "old" data of DB12 is overwritten with the data of DB14.

The data block is configured to overwrite 60 values (1 minute) with "zero".

DB15, BatchProtocol

DB15 contains the data for a "batch protocol".

In this case the data is stored permanently and serves as "sample application".

DB100, InstanceDB_FB100

Instance DB for FB100

DB101, InstanceDB_FB101

Instance DB for FB101

DB102, InstanceDB_FB102

Instance DB for FB102

DB110, StorageLocationName

DB110 contains the name of the "*.csv file" to be exported created by the STEP 7 program.

DB112, AdditionalCharacter

DB120 contains characters needed for the file name.

- „**0**“ (to put a "zero" before a number)
- „**_**“ (to put an underscore before a number)
- „**.csv**“ (to add a file name at the end of the file identification)

4 Using the Sample Application

This chapter describes how to use the application.

Requirements

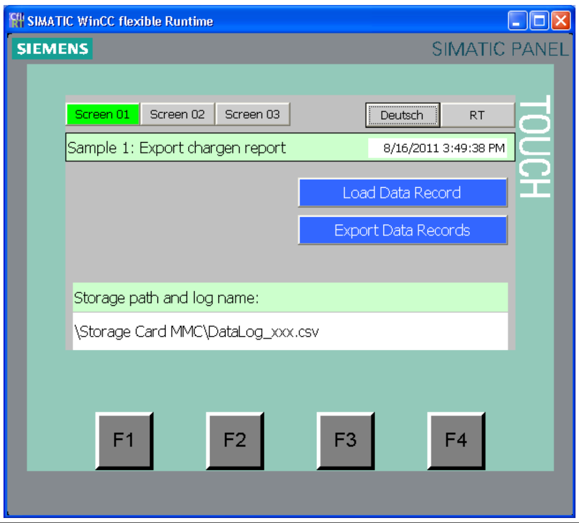
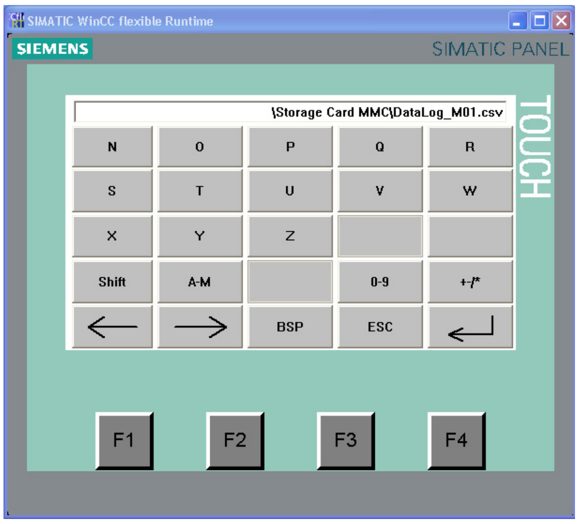
- S7 controller
- TP177B PNDP with MMC memory card for storing the exported "*.csv files".

Alternatively, you can also test the attached configuration with PLCSIM and the WinCC flexible Runtime.

4.1 Example 1

Read out and export batch data.

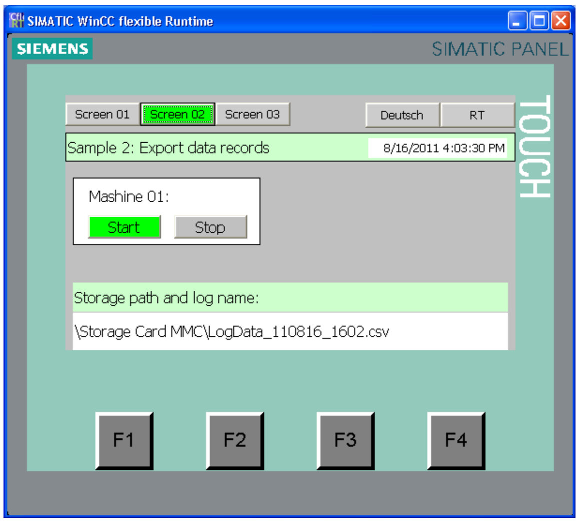
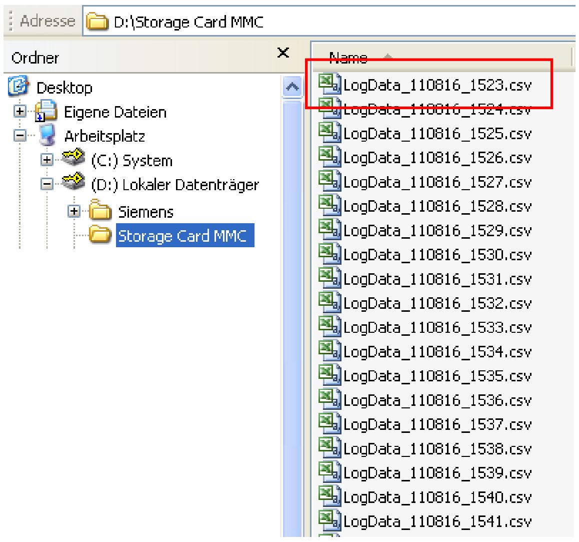
Table 4-1

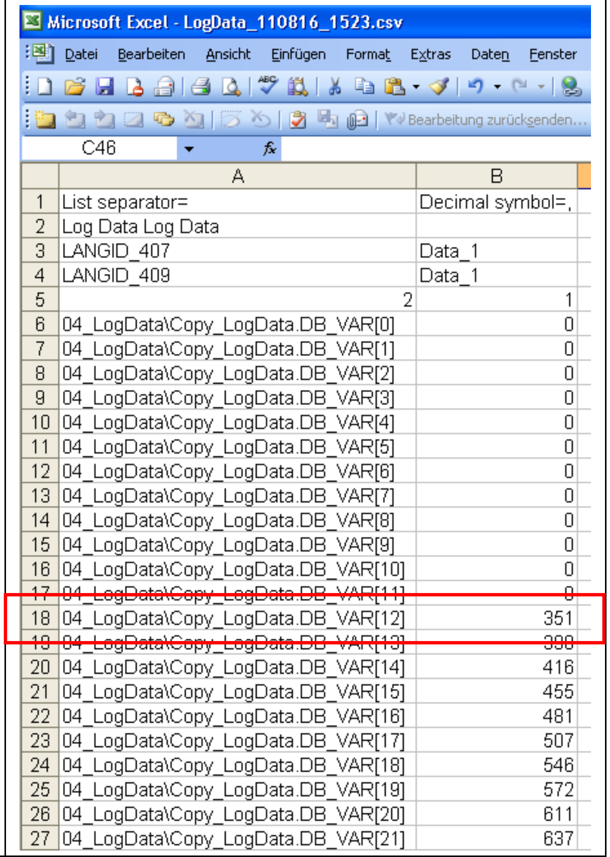
No.	Description	Pictures
1.	**Screen 1** **Import and export data:** Click the "**Load Data Record**" button to read out the permanently stored data record (batch protocol) from the S7 controller. Click the "**Export Data Record**" button to export the loaded data record (batch protocol). The exported file is given the name specified in the text field.	 SIMATIC WinCC flexible Runtime SIEMENS SIMATIC PANEL Screen 01 Screen 02 Screen 03 Deutsch RT TOUCH Load Data Record Export Data Records F1 F2 F3 F4 Sample 1: Export chargen report 8/16/2011 3:49:38 PM Storage path and log name: \Storage Card MMC\DataLog_xxx.csv
2.	**Screen 1** **Define the name of the export file:** Click the IO text field to open the system keyboard of the operator panel. The file name is preset with a "**Default name**" (\Storage Card MMC\DataLog_xxx.csv). In this way, the operator only has to change the "**DataLog_xxx**" name accordingly.	 SIMATIC WinCC flexible Runtime SIEMENS SIMATIC PANEL TOUCH N O P Q R S T U V W X Y Z Shift A-M 0-9 +-/* BSP ESC F1 F2 F3 F4 \Storage Card MMC\DataLog_M01.csv

4.2 Example 2

Automatic archiving of machine data and export as "*.csv file".

Table 4-2

No.	Description	Pictures
1.	**Screen 2** Click "**Start**" button to start measurement of the machine data. • A "simulated value" is stored every second in a DB. • A "*.csv file is created automatically at the beginning of a new minute. The file contains the data collected up to that point in time. Click "**Stop**" button to end measurement of the machine data. The text output field displays the file name of the file last saved. In this example the last file was created on August 16, 2011 at 4:02 PM.	 SIMATIC WinCC flexible Runtime SIEMENS SIMATIC PANEL Screen 01 Screen 02 Screen 03 Deutsch RT TOUCH Start Stop F1 F2 F3 F4 Sample 2: Export data records 8/16/2011 4:03:30 PM Mashine 01: Storage path and log name: \Storage Card MMC\LogData_110816_1602.csv
2.	**Screen 2** **View of "*.csv files" created:** In this example the last file was created on August 16, 2011 at 3:23 PM *). The subsequent files were each created at the beginning of a new minute. ***)** See also the points below.	 Adresse D:\Storage Card MMC Ordner Desktop Eigene Dateien Arbeitsplatz (C:) System (D:) Lokaler Datenträger Siemens Storage Card MMC Name LogData_110816_1523.csv LogData_110816_1524.csv LogData_110816_1525.csv LogData_110816_1526.csv LogData_110816_1527.csv LogData_110816_1528.csv LogData_110816_1529.csv LogData_110816_1530.csv LogData_110816_1531.csv LogData_110816_1532.csv LogData_110816_1533.csv LogData_110816_1534.csv LogData_110816_1535.csv LogData_110816_1536.csv LogData_110816_1537.csv LogData_110816_1538.csv LogData_110816_1539.csv LogData_110816_1540.csv LogData_110816_1541.csv

No.	Description	Pictures
3.	**Screen 2** **Content of the "*.csv file":** The figure shows the contents of the "LogData_110816_1523.csv" file. The "Index" displayed • DB_VAR[**0**] • DB_VAR[**1**] • DB_VAR[**2**] • etc. indicates the current "**second**" at which the first value was archived. In this example, archiving of the measured values began on August 16, 2011 at 3:23:**12** PM.	

Microsoft Excel - LogData_110816_1523.csv

	A	B
1	List separator=	Decimal symbol=,
2	Log Data Log Data	
3	LANGID_407	Data_1
4	LANGID_409	Data_1
5	2	1
6	04_LogData\Copy_LogData.DB_VAR[0]	0
7	04_LogData\Copy_LogData.DB_VAR[1]	0
8	04_LogData\Copy_LogData.DB_VAR[2]	0
9	04_LogData\Copy_LogData.DB_VAR[3]	0
10	04_LogData\Copy_LogData.DB_VAR[4]	0
11	04_LogData\Copy_LogData.DB_VAR[5]	0
12	04_LogData\Copy_LogData.DB_VAR[6]	0
13	04_LogData\Copy_LogData.DB_VAR[7]	0
14	04_LogData\Copy_LogData.DB_VAR[8]	0
15	04_LogData\Copy_LogData.DB_VAR[9]	0
16	04_LogData\Copy_LogData.DB_VAR[10]	0
17	04_LogData\Copy_LogData.DB_VAR[11]	0
18	04_LogData\Copy_LogData.DB_VAR[12]	351
19	04_LogData\Copy_LogData.DB_VAR[13]	390
20	04_LogData\Copy_LogData.DB_VAR[14]	416
21	04_LogData\Copy_LogData.DB_VAR[15]	455
22	04_LogData\Copy_LogData.DB_VAR[16]	481
23	04_LogData\Copy_LogData.DB_VAR[17]	507
24	04_LogData\Copy_LogData.DB_VAR[18]	546
25	04_LogData\Copy_LogData.DB_VAR[19]	572
26	04_LogData\Copy_LogData.DB_VAR[20]	611
27	04_LogData\Copy_LogData.DB_VAR[21]	637

4.3 Tips and Tricks

No data is archived on the memory card

- Check the storage location you are using on your operator panel with the one you have configured (USB, SD card, MMC card, etc.).
- For "Example 2", check the "status" of the system functions "Get data record from PLC" and "Export data records".
  You can see the status in "**Screen 3**" of the HMI configuration and if necessary, reset it using the "Reset" button.
- Archiving can only be done automatically if the operator panel's Runtime is running.

There is a "2" as file name in the text output field of Example 2

"2" is output as file name in the text output field before the first execution of the "export function" in Example 2.

This depends on the data type used and does not affect the function. If this disturbs you, you can preset the tag with a "default value" via the controller program.

Edit file name

The file name is "compiled" in FB102.

In the configuration phase it is useful to create a "variable table" (see "VAT_StoragePathName").

Via the "variable table" you can quickly see as of which "byte" a new "name" begins.

Number of process values that can be archived

In this sample application the number of values to be archived depends on the number of entries you can have in a **recipe**.

The number of entries depends on the system limit of the operator panel used.

Refer here to the Online Help of WinCC flexible.
Keyword: "system limits".

Archiving time

In this sample application the values are archived every second over a period of one minute.

You can change the time period to suit your requirements.

- Second-trigger: In this sample application via OB35
- Minute evaluation: read out and evaluation of the current CPU time.

Refer here to the Online Help of STEP 7.
Keyword: "Format of the DATE_AND_TIME data type".

There is a table there which describes the content of the bytes that contain date and time data.

Simulated "measured values"

The "measured values" for "**Example 2**" are generated with the "**SimulateTag**" function.

The function is configured in the Properties of the separate screens (Events > Loaded).

If you specify "real" values, you no longer need this function. Transfer the relevant value to the tag "DB10.DBW2" (**ProgData.PressureMeasurement**).