SIEMENS

SIMATIC S7

Product Information

A5E00169432-02

Edition 07/2003

Manual S7 Distributed Safety, Configuring and Programming A5E00109537-01

Scope

This product information document supplements the *S7 Distributed Safety, Configuring and Programming* manual, A5E00109537-01, Edition 03/2002.

Documentation Package

The manual indicated above and this product information document are included in the documentation package *S7 Distributed Safety*, 6ES7 988-8FB10-8BA0.

Organization of Product Information Document

This product information document is organized in two parts. The first part describes all the changes to the optional package *S7 Distributed Safety*, V 5.2 + Service Pack 1 compared with Version V 5.1.

The second part presents corrections to the S7 Distributed Safety, Configuring and Programming manual, A5E00109537-01, Edition 03/2002, that could not be made prior to publication. These corrections apply to versions V 5.1, V 5.2 and V 5.2 + Service Pack 1 of the *S7 Distributed Safety* optional package.

Cross-References in this Product Information

For the sake of brevity, references to sections of the manual mentioned above do not include the name of the manual (for example, "see manual, Section *6.3*").

All cross-references that do not indicate a specific publication refer to this product information documentation (for example, "see Section *1.2.2*").

Siemens Aktiengesellschaft

Contents

1 Converting from *S7 Distributed Safety*, V 5.1 to V 5.2 + Service Pack 1

What's New in *S7 Distributed Safety*, V 5.2 Compared with V 5.1?

The major innovations in *S7 Distributed Safety*, V 5.2 compared with V 5.1 are listed below:

- Support for F-CPUs IM 151-7 F-CPU and CPU 416F-2
- Safety-oriented master to I-slave communication
- STEP 7 operations JMP, JMPN, RET, and OV in F-FBD/F-LAD
- Length of signatures of the safety program = 32 bits (collective signature of all F-blocks with an F-attribute in the block container; collective signature of the safety program; signature of the symbols)
- "Check block consistency" function
- F-application block F_SCA_I: scaling of values of data type INT
- Enhanced functionality of the "Compare Program" dialog box
- Expanded printout of safety program project data

What's New in *S7 Distributed Safety*, V 5.2 + Service Pack 1 compared with V 5.2?

The major innovations in *S7 Distributed Safety*, V 5.2 + Service Pack 1 compared with V 5.2 are listed below:

- Support of F-CPU 317F-2 DP
- Safety-oriented I-slave to I-slave communication
- Modification of automatically assigned PROFIsafe target addresses in *HW Config*
- Know-how protection for F-FBs and F-FCs written by the user
- F application block F_INT_WR: Writing a value of the data type INT indirectly to an F-DB
- F application block F_INT_RD: Reading a value of the data type INT indirectly from an F-DB
- F application block F_1oo2DI: 1oo2 evaluation with Discrepancy Analysis
- Support of F libraries created by the user

Software Requirements for *S7 Distributed Safety*, V 5.2 + Service Pack 1

The following software package (at least) must be installed on the PC or programming device:

STEP 7, V 5.1 + Service Pack 6 or higher or:

Please note that use of the following functions is possible only with higher *STEP 7* versions:

Function	Required *STEP 7* Version
Modification of automatically assigned PROFIsafe target addresses in *HW Config*	
F submodules ET 200S	*STEP 7*, V 5.2
Fail-safe DP standard slaves	*STEP 7*, V 5.2 + Service Pack 1
Using F libraries created by the user	*STEP 7*, V 5.2
Using safety-oriented I-slave to I-slave communication	*STEP 7*, V 5.2 + Service Pack 1
Using a CPU 317F-2 DP	*STEP 7*, V 5.2 + Service Pack 1

Safety Note

The use of the optional package *S7 Distributed Safety*, V 5.2 + Service Pack 1 with earlier versions of *STEP 7* is not permitted.

Software Requirements for Configuring F-CPUs

The following software is required to configure F-CPUs for use in *S7 Distributed Safety*:

F-CPU	As of Order No.	*S7 Distributed Safety*	*STEP 7*
IM151-7 F-CPU	6ES7 151-7FA00-0AB0	V 5.2	V 5.2
CPU 315F-2 DP	6ES7 315-6FF01-0AB0	V 5.2	V 5.1 + Service Pack 6
CPU 416F-2	6ES7 416-2FK02-0AB0	V 5.2	V 5.2
CPU 317F-2 DP	6ES7 317-6FF00-0AB0	V 5.2 + Service Pack 1	V 5.2 + Service Pack 1

Calculation of Maximum Response Time of Your F-System

Use the Microsoft Excel file provided with *S7 Distributed Safety*, V 5.**2 + Service Pack 1 (SP 1)** to calculate the maximum response time of your F-system.

Note

When you install *S7 Distributed Safety*, V 5.2 + SP 1, the Excel file for V 5.1 or. V 5.2 (*...\Siemens\STEP7\S7Manual\s7fco\s7fcotib.xls)* supplied with the optional package is overwritten. If you made your response time calculations directly in this file rather than in a copy of the file that you created in a **different** folder, save the V 5.1 or V 5.2 file in another folder **before *installing S7 Distributed Safety*, V 5.2 + SP 1**.

Otherwise your calculations in V 5.1 or V 5.2 will be lost when you install *S7 Distributed Safety*, V 5.2 + SP 1!

If you want to update the calculations for *S7 Distributed Safety*, V 5.2 + SP 1, transfer these entries manually to the V 5.2 + SP 1 file.

Conversion of *S7 Distributed Safety* from V 5.1 to V 5.2 + SP 1

Reading a Safety Program with *S7 Distributed Safety* V 5.2 + SP 1**:**

If you would like to use *S7 Distributed Safety* V 5.2 +SP 1 to read, but not change, a safety program created with *S7 Distributed Safety* V 5.1, open the "Safety Program" dialog box with V 5.2 +SP 1 . Do **not** compile the safety program.

Note

When you open the "Safety Program" dialog, of a consistent safety program created with *S7 Distributed Safety*, V 5.1, the status: "The safety program is consistent." is displayed although different signatures are displayed.

This is due to the fact that the length of the signatures has been changed from 16 to 32 bits.

Modifying a Safety Program with *S7 Distributed Safety* V 5.2 + SP 1:

If you want to use *S7 Distributed Safety* V 5.2 + SP 1 to change a safety program created with V 5.1, proceed as follows:

1. Compile the safety program with *S7 Distributed Safety* V 5.2 + SP 1 prior to making changes.

   Result: All F-blocks of the *Distributed Safety* library (V1) that were used in the safety program and for which there is a new version in the *Distributed Safety* library of V 5.2 + SP 1 are **automatically** replaced following confirmation. The collective signature of all F-blocks and the signature of individual F-blocks change for the following reasons:

   - Length of collective signature has been changed from 16 to 32 bits
   - F-blocks of the *Distributed Safety* (V1) F-library were replaced
   - Automatically compiled F-blocks have changed
2. Change the safety program according to your requirements.
3. Recompile the safety program.
4. Perform a comparison of the old and new version of the safety program.
   - You can identify changes to the version of an F-block of the *Distributed Safety* F-library (V1) by the changes to F-block signatures. Modified signatures and initial value signatures of all F-application blocks and F-system blocks must conform to those in Annex 1 of the Certification Report.
   - Furthermore, you can identify whether changes have been made in the safety program. If necessary, the safety program must undergo another acceptance test (see manual, Section *6.3*).

Conversion of S7 Distributed Safety from V 5.2 + SP 1 to V 5.1

Safety Note

A safety program compiled with *S7 Distributed Safety* V 5.2 + SP 1 must not be read or edited with *S7 Distributed Safety* V 5.1.

1.1 Configuration

1.1.1 Configuring Safety-Related Master to I-Slave Communication

Overview

Note

This section is only relevant if you are using *S7 Distributed Safety* V 5.2 + SP 1 with ***STEP 7*, ≤ V 5.2**.

If you use *STEP 7*, ≥ V 5.2 + SP 1, the information in Sections 1.1.2 through 1.1.4 is relevant to you.

The manual describes safety-related master to master communication.

The following section describes the configuration of an additional safety-related communication option, that is, safety-related communication between safety programs in different F-CPUs. As in a standard system, safety-related communication between the safety program of the F-CPU of a DP master and the safety program(s) of the F-CPU(s) of one or more intelligent DP slaves takes place by means of master to I-slave connections.

You do not need any additional hardware for the master to I-slave communication.

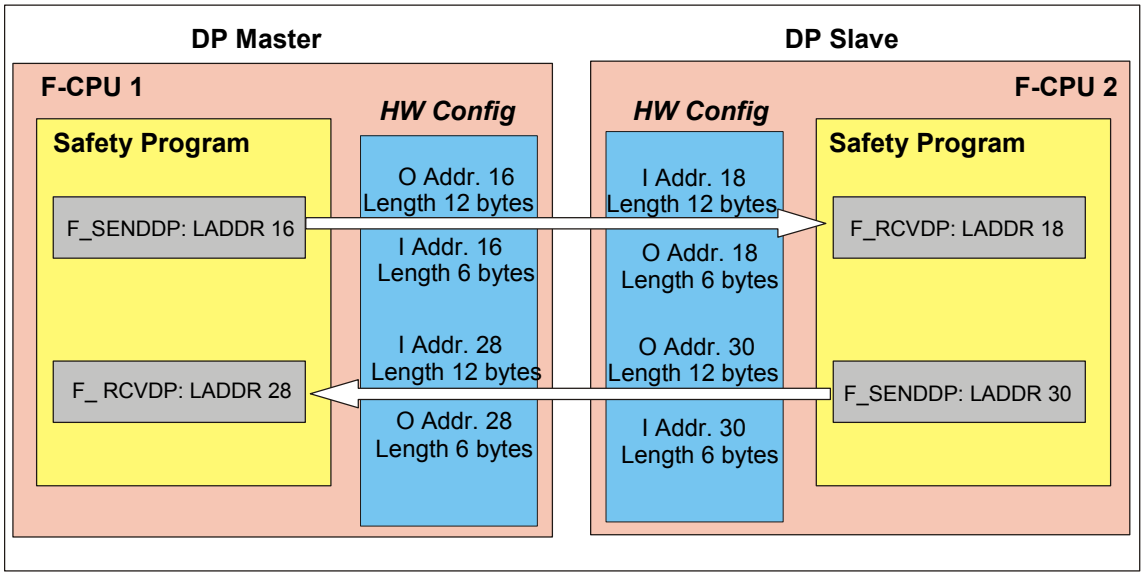
Communication by Means of F_SENDDP and F_RCVDP

Safety-related communication occurs by means of the F-application blocks F_SENDDP and F_RCVDP that you used in the safety programs of the F-CPUs.

Configuring Input/Output Data Areas

You must configure both an output data area and an input data area in *HW Config* for each communication connection between two F-CPUs. In the figure below, each of the two F-CPUs is supposed to be able to send and receive data. You must therefore configure two output data areas and two input data areas for each F-CPU.

You assign the configured start addresses of the input and output data areas to the LADDR parameter of the corresponding
F-application blocks F_SENDDP and F_RCVDP in the safety programs (see manual, Section *5.4*).



Rules for Defining the Data Areas

The output data area for the **data to be sent** must begin with the same start address as the associated input data area. A total of 12 bytes (consistent) must be configured for the output data area, and 6 bytes (consistent) for the input data area.

The input data area for the **data to be received** must begin with the same start address as the associated output data area. A total of 12 bytes (consistent) must be configured for the output data area, and 6 bytes (consistent) for the input data area.

Procedure for Configuring Master to I-Slave Communication

The same procedure is used to configure safety-related communication master to I-slave communication as is used to configure master to I-slave communication in a standard system.

The figure below illustrates the configuration procedure for the address areas in the previous figure.

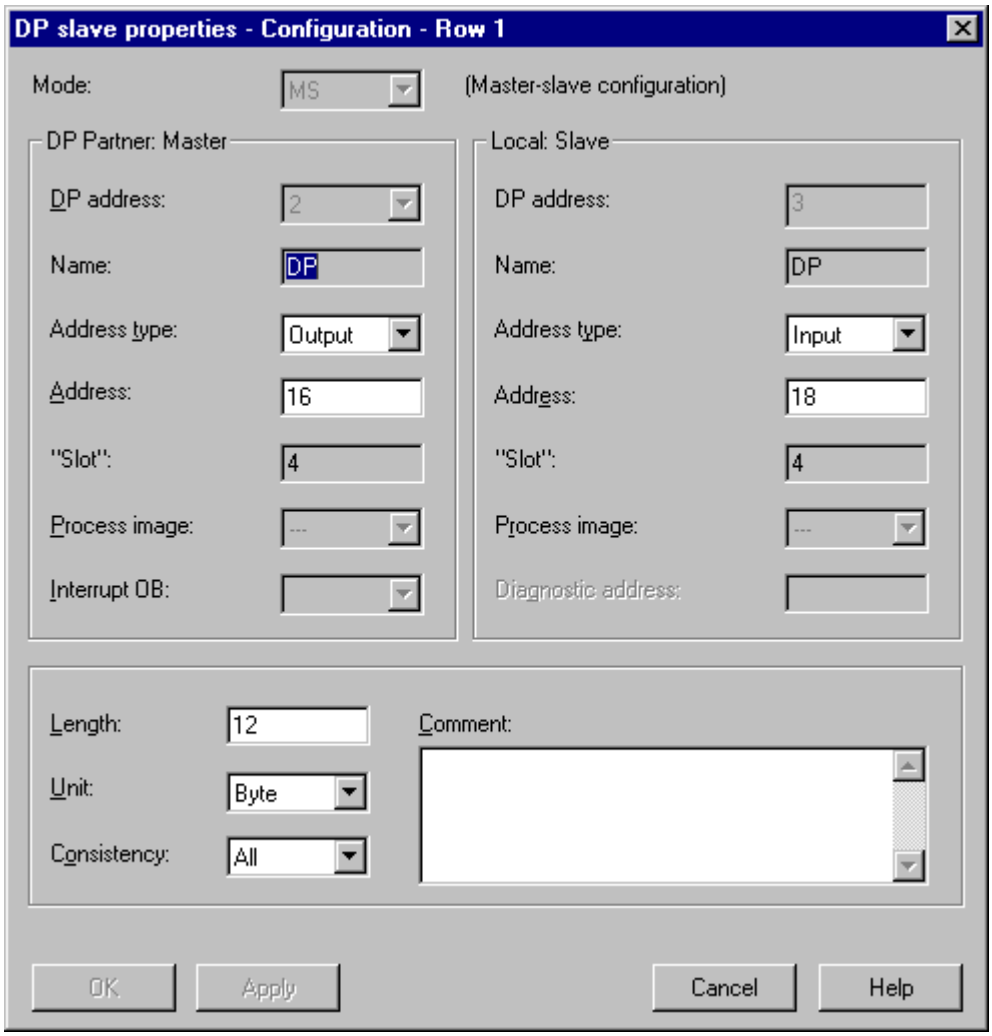
Requirement:

You have created a project in *STEP 7*.

1. Create a station in your project (in SIMATIC Manager, for example, an S7-300 station).
2. Assign a CPU with fail-safe capability to this station (in *HW Config*, from the hardware catalog).
3. Configure this CPU as a DP slave (in *HW Config*, in the Object Properties of the DP interface of the CPU).
4. Create another station and assign a CPU with fail-safe capability (see steps 1 and 2).
5. Configure this CPU as a DP master (in *HW Config*, in the Object Properties of the DP interface of the CPU).
6. In the hardware catalog, under "Configured stations," select the station type of the intelligent DP slave ("CPU 31x" in this example) and position it on the DP master system.
7. Link the intelligent DP slave to the DP master in the Connection dialog box, which is displayed automatically.
   Now, you can specify the input and output data areas for the safety-related master to slave communication:
8. In the "Configuration" tab of the Object Properties of the intelligent DP slave, select "New."
9. Enter an output data area for the DP master and the associated input data area for the intelligent DP slave. For this example, make the following entries:

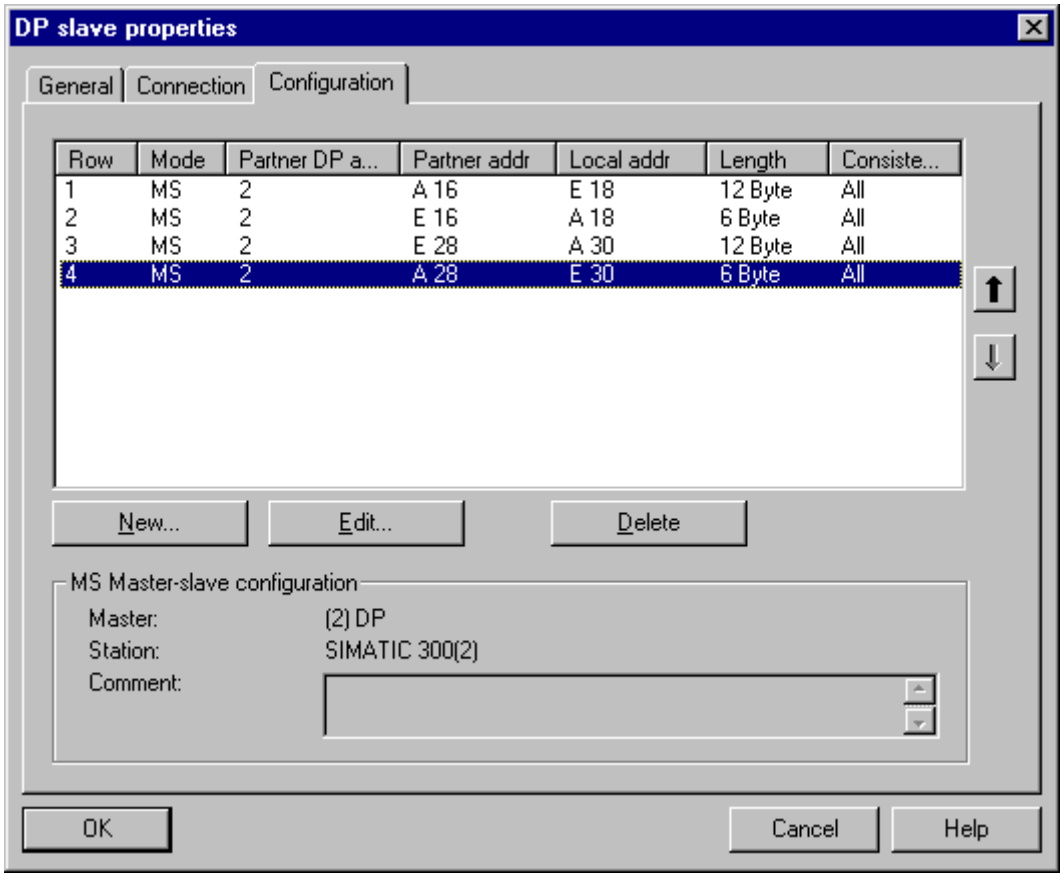
   - For "DP partner: Master", enter "Output" for Address type, "16" for Address, "12" for Length, "Byte" for Unit, and "Total" for Consistent.
   - For "Local: Slave", enter "Input" for Address type and "18" for Address.

   The dialog box has the following appearance:



10. Confirm your entries with "OK."

11. Continue following steps 8 and 9 until all output and input data areas are defined.
    This results in four configuration rows for this example:



Note

Be sure to use identical values for the start addresses of the output and input data areas.

A total of 12 bytes (consistent) must be configured for the output data area, and 6 bytes (consistent) for the input data area.

Always select the "Consistency: All" option for all input and output data areas.

Additional Information

For information on programming safety-related master to I-slave communication, refer to Section *1.2.6*.

For information on master to I-slave communication, refer to the *STEP 7 online help*.

1.1.2 Safety-Related CPU-CPU Communication

Note

This section and the following sections 1.1.3 and 1.1.4 are valid only if you are using *S7 Distributed Safety* V 5.2 + SP 1 with ***STEP 7*, ≥ V 5.2 + SP 1**.

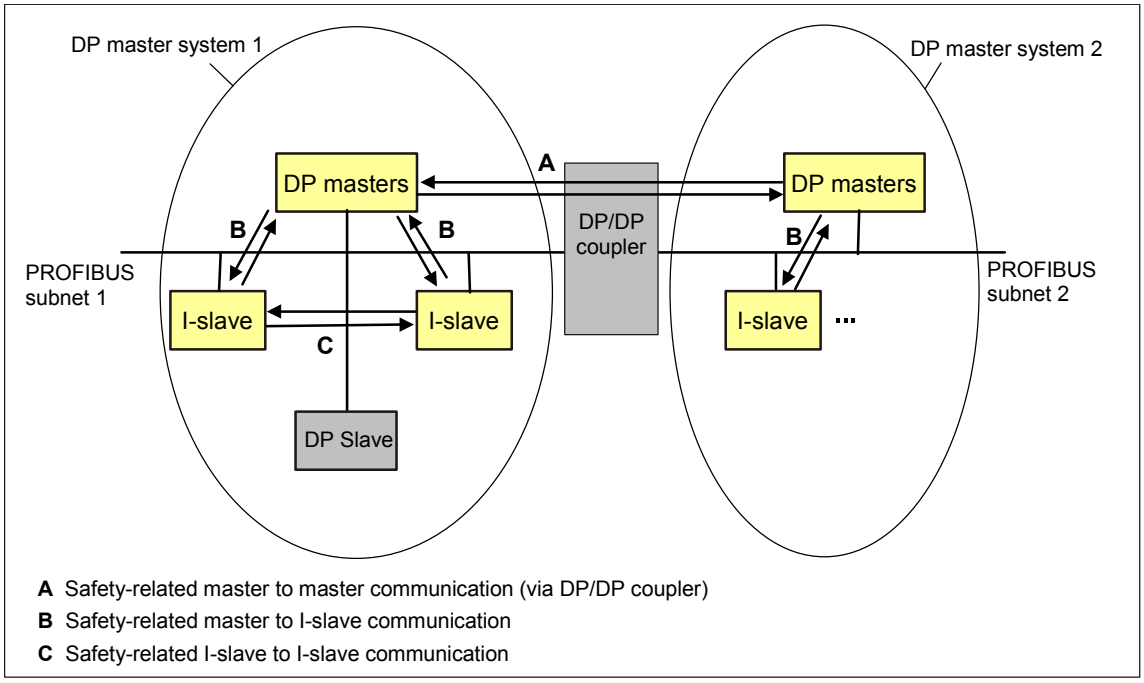
If you are using *STEP 7*, ≤ V 5.2, the information in Section 1.1.1 applies.

Overview

The schematic below shows you an overview of the 3 options for safety-related CPU-CPU communication in S7 Distributed Safety F-systems.

In safety-related CPU-CPU communication, a fixed amount of fail-safe data of the data types BOOL and INT is transferred fail-safe between the safety programs in F-CPUs of DP masters/intelligent DP slaves.

The data transfer makes use of F-application blocks F_SENDDP for sending and F_RCVDP for receiving. The data is stored in configured address areas of the DP master/intelligent DP slave (I-slave).



A Safety-related master to master communication (via DP/DP coupler)
B Safety-related master to I-slave communication
C Safety-related I-slave to I-slave communication

Further Information

Safety-related master to master communication is described in the Manual.

Safety-related master to I-slave communication and safety-related I-slave to I-slave communication are described in the following sections of this product information.

1.1.3 Configuring Safety-Related Master to I-Slave Communication

Overview

Safety-related master to master communication is described in the manual.

The following section describes the configuration of an additional safety-related communication option, that is, safety-related communication between safety programs in different F-CPUs. As in a standard system, safety-related communication between the safety program of the F-CPU of a DP master and the safety program(s) of the F-CPU(s) of one or more intelligent DP slaves (I-slaves) is handled over master to I-slave connections.

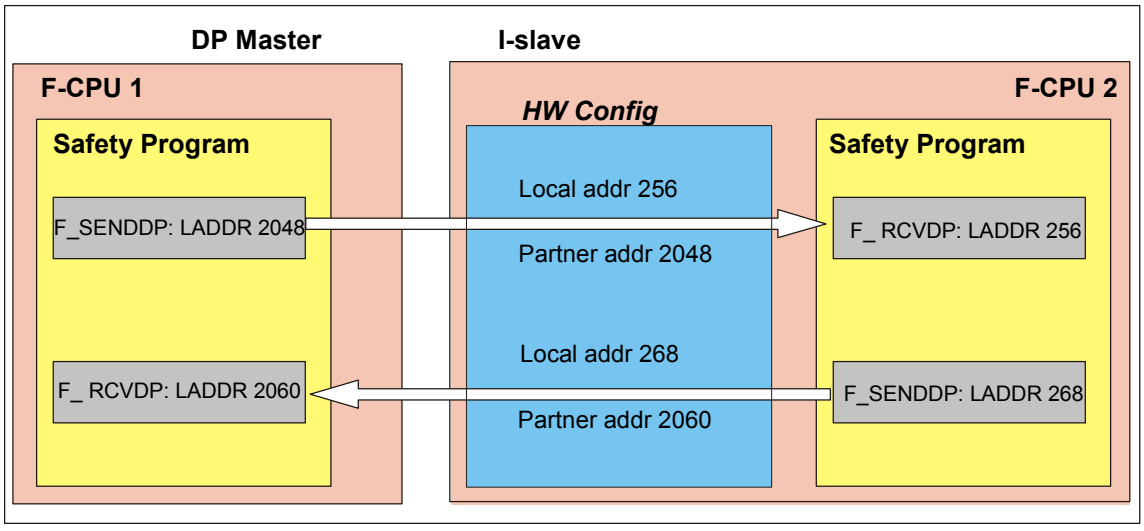
You do not need any additional hardware for the master to I-slave communication.

Communication by Means of F_SENDDP and F_RCVDP

Safety-related communication occurs by means of the F-application blocks F_SENDDP and F_RCVDP that you used in the safety programs of the F-CPUs.

Configuring Address Areas

For every communication connection between two F-CPUs, you must configure address areas in *HW Config*. In the figure below, each of the two F-CPUs is supposed to be able to send and receive data.



You specify the configuration of the following in the object properties dialog of the I-slave:

- to send to the DP master, a local address (I-slave) and a partner address (DP master)
- to receive from the DP master, a local address (I-slave) and a partner address (DP master)

You assign the configured addresses to the LADDR parameter of the corresponding F application blocks F_SENDDP and F_RCVDP in the safety programs (see Section *1.2.6* and manual, Section *5.4*).

Address Areas

Each of the local partner addresses represents a start address of an address area of input and output data. After configuring the local and partner addresses, the address areas are automatically assigned. The assigned address areas for a send and a receive connection are shown in the following table:

Communication Connection	**Assigned Address Area on the F-CPU of the ...**
Send: I-slave to DP master	I-slave: 12 bytes of output and 6 bytes of input data
	DP master: 12 bytes of input and 6 bytes of output data
Receive: I-slave from DP master	I-slave: 12 bytes of input and 6 bytes of output data
	DP master: 12 bytes of output and 6 bytes of input data

Note

We recommend that you use addresses outside the process image as the local and partner addresses, since the process image should be reserved for the address areas of modules. When configuring the address areas, the next free address outside the process image is therefore proposed.

How to Configure Master to I-Slave Communication

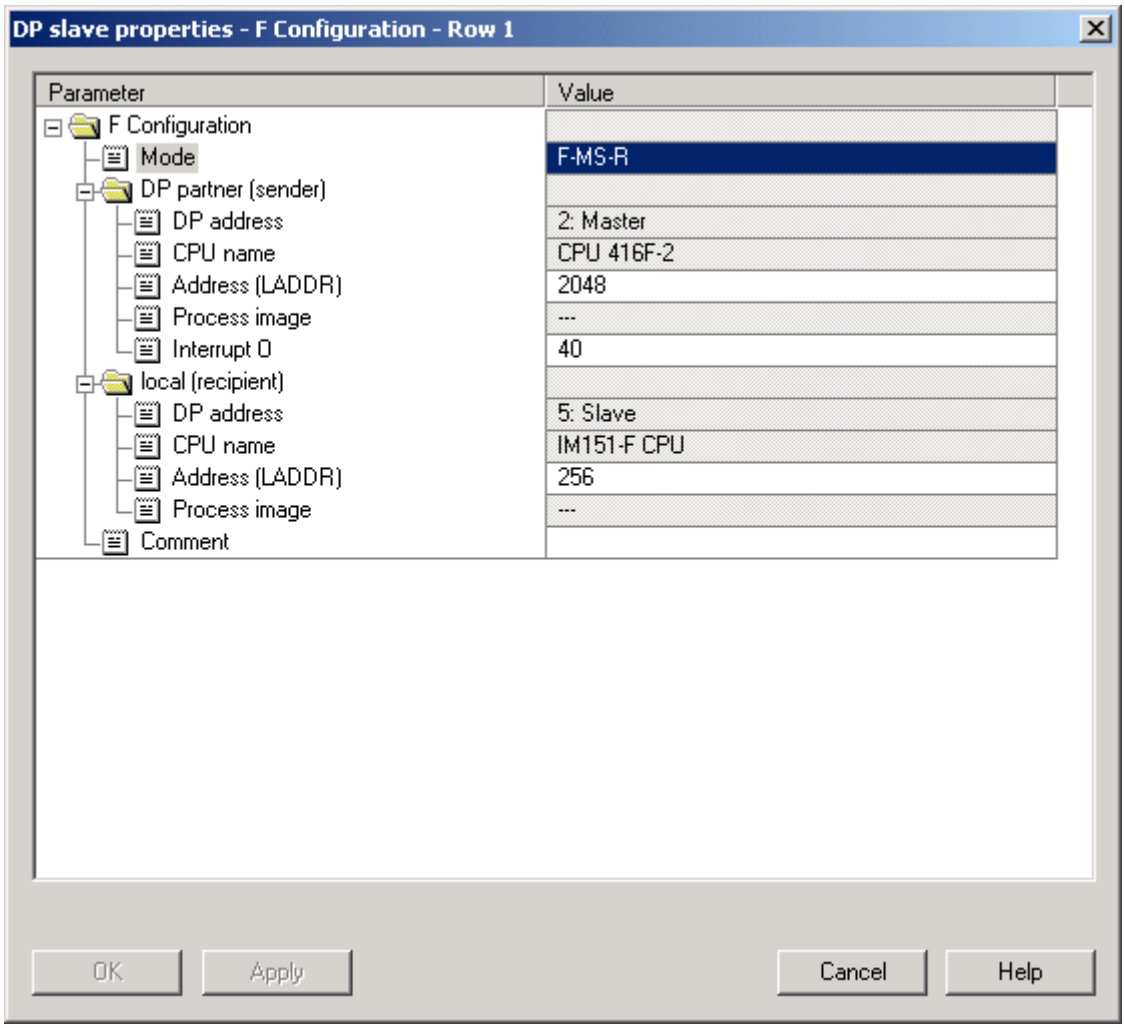
The figure below illustrates the configuration procedure for the address areas in the previous figure.

Requirement:

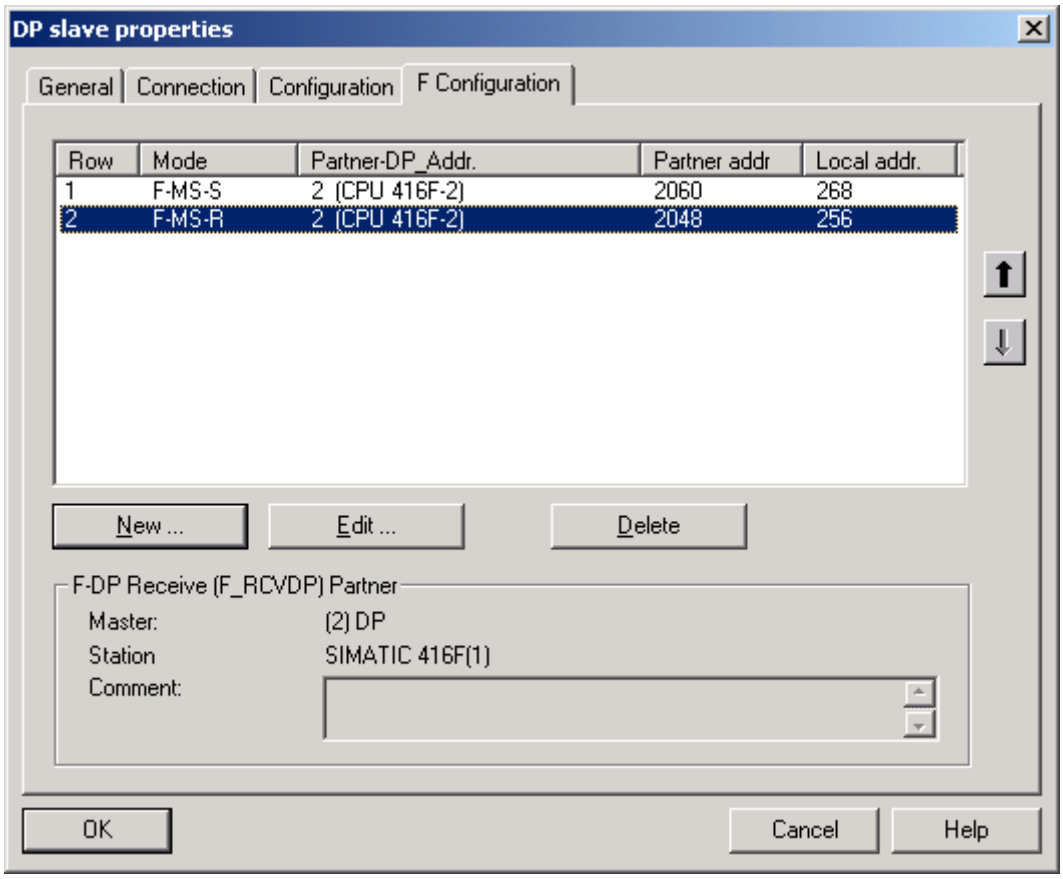
You have created a project in *STEP 7*.

1. Create a station in your project (in *SIMATIC Manager,* for example, an S7-300 station).
2. Assign a CPU with fail-safe capability to this station (in *HW Config*, from the hardware catalog).
3. Configure this CPU as a DP slave (in *HW Config*, in the "Operating Mode" tab of the Object Properties for the DP interface of the CPU).
4. Create another station and assign a CPU with fail-safe capability (see steps 1 and 2).
5. Configure this CPU as a DP master (in *HW Config*, in the "Operating Mode" tab of the Object Properties for the DP interface of the CPU).
6. In the hardware catalog, under "Configured stations," select the station type of the I-slave (for example, the "CPU 31x") and place it on the DP master system.
7. Link the I-slave to the DP master in the Connection dialog box that opens automatically.
   You can now specify the address areas for the safety-related master to I-slave communication:
8. In the "F-Configuration" tab of the Object Properties of the I-slave, select "New."
9. In the next dialog, make the following entries for the receive connection from the DP master for our example:
   - For "Mode: F-MS-R" (receive over a fail-safe master to I-slave communication)
   - For "DP partner (sender): address (LADDR): 2048"
   - For "local (receiver): address (LADDR): 256"
   - Accept the defaults for the other parameters in the dialog.

   The dialog appears as shown below:



10. Confirm your entries with "OK."
11. In the "F-Configuration" tab of the Object Properties of the I-slave, select "New."
12. In the next dialog, make the following entries for the send connection to the DP master for our example:
    - For "Mode: F-MS-S" (send over a fail-safe master to I-slave communication)
    - For "DP partner (receiver): address (LADDR): 2060"
    - For "local (sender): address (LADDR): 268"
13. Confirm your entries with "OK."
    This results in two configuration rows for this example:



Note

In the object properties of the I-slave, entries are automatically made in the "Configuration" tab based on the configuration in the "F-Configuration" tab. These entries must not be modified. Otherwise, safety-related master to I-slave communication is not possible.

The assigned address areas on the DP master and I-slave can be seen in the "Configuration" tab.

Additional Information

You will find a description of the parameters in the *context-sensitive online help of the "F-Configuration" tab*.

For information on programming safety-related master to I-slave communication, refer to Section *1.2.6*.

For information on master to I-slave communication, refer to the *STEP 7 online help*.

For information on the address areas, process image partitions, and supported interrupt OBs, refer to the Technical Specifications of the F-CPU you are using.

1.1.4 Configuring Safety-Related I-Slave to I-Slave Communication

Overview

The manual describes safety-related master to master communication.

The following section describes the configuration of an additional safety-related communication option, that is, safety-related communication between safety programs in different F-CPUs. As in standard systems, safety-related communication between the safety program of the F-CPUs of intelligent DP slaves involves direct data exchange.

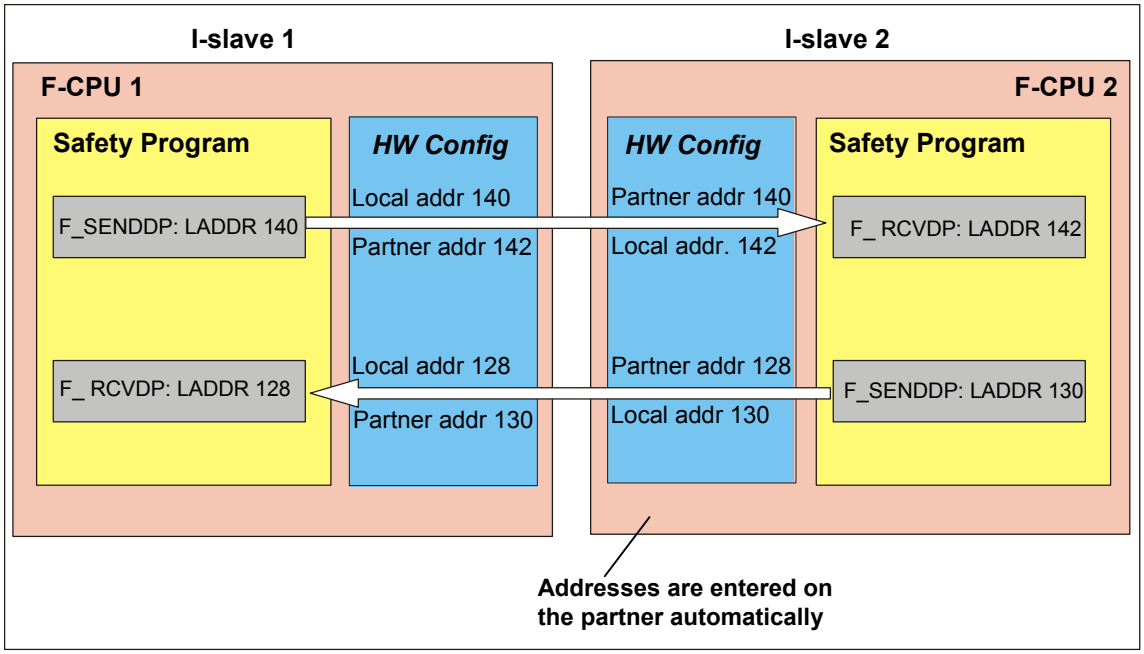
You do not need any additional hardware for I-slave to I-slave communication.

Communication by Means of F_SENDDP and F_RCVDP

Safety-related communication occurs by means of the F-application blocks F_SENDDP and F_RCVDP that you used in the safety programs of the F-CPUs.

Configuring Address Areas

For every communication connection between two F-CPUs, you must configure address areas in *HW Config*. In the figure below, each of the two F-CPUs is supposed to be able to send and receive data.



You specify the configuration of the following in the object properties dialog of I-slave 1:

- to send to I-slave 2, a local address (I-slave 1) and a partner address (I-slave 2)
- to receive from I-slave 2, a local address (I-slave 1) and a partner address (I-slave 2)

No further configuration of communication is necessary in the object properties dialog of I-slave 2. The addresses are entered automatically in the object properties dialog of I-slave 2.

You assign the configured addresses to the LADDR parameter of the corresponding F application blocks F_SENDDP and F_RCVDP in the safety programs (see Section *1.2.6* and manual, Section *5.4*).

Address Areas

Each of the local partner addresses represents a start address of an address area of input and output data. After configuring the local and partner addresses, the address areas are automatically assigned. The assigned address areas for a send and a receive connection are shown in the following table:

Communication Connection	**Assigned Address Area on the F-CPU* of the ...**
Send: I-slave 1 to I-slave 2	I-slave 1: 12 bytes of output and 6 bytes of input data
	I-slave 2: 12 bytes of input and 6 bytes output data
	DP master: 12 + 6 bytes of input data
Receive: I-slave 1 from I-slave 2	I-slave 1: 12 bytes of input and 6 bytes of output data
	I-slave 2: 12 bytes of output and 6 bytes of input data
	DP master: 12 + 6 bytes of input data

* The CPU of the DP master can be an F-CPU or a standard CPU.

Note

We recommend that you use addresses outside the process image as the local and partner addresses, since the process image should be reserved for the address areas of modules. When configuring the address areas, the next free address outside the process image is therefore proposed.

How to Configure I-Slave to I-Slave Communication

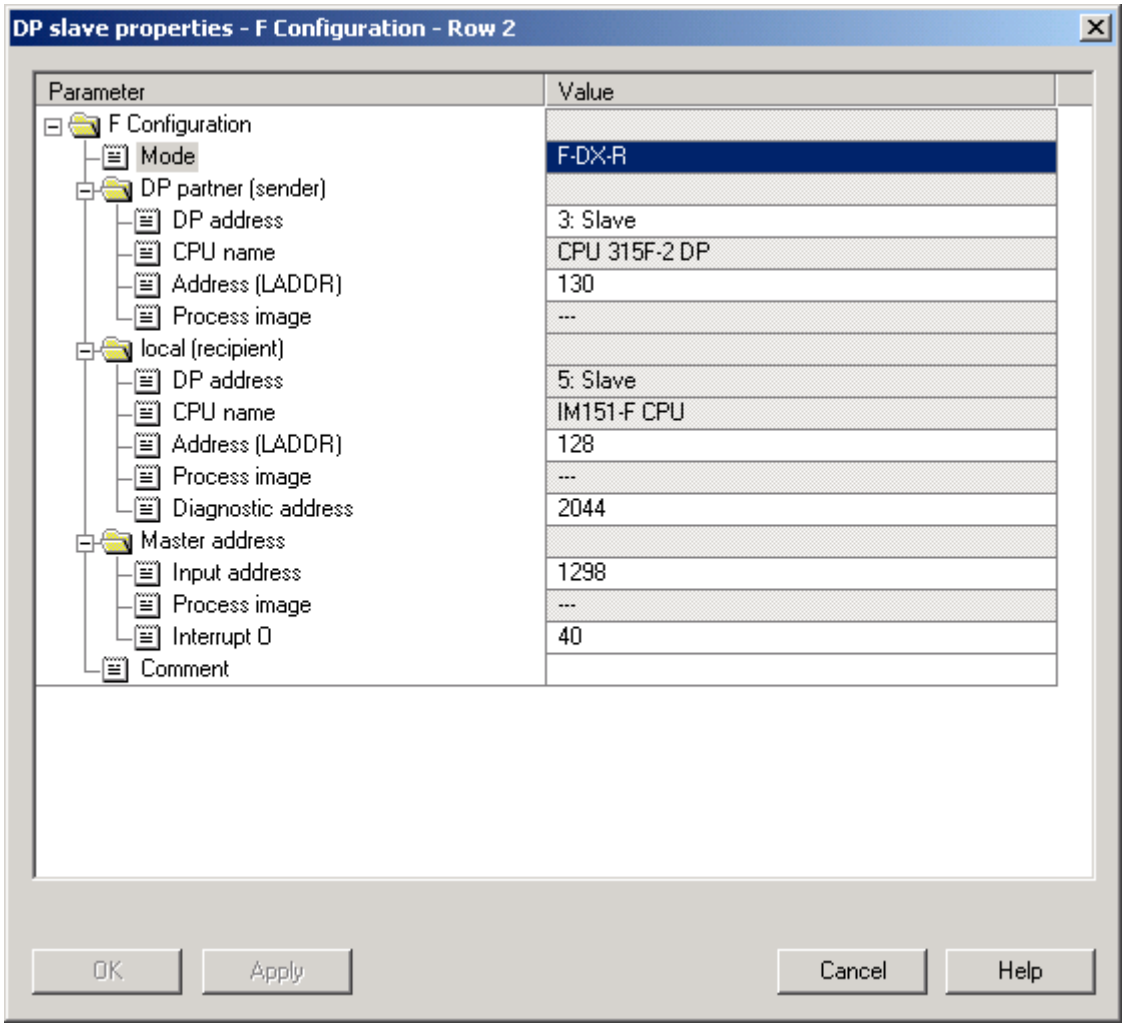
The figure below illustrates the configuration procedure for the address areas in the previous figure.

Requirement:

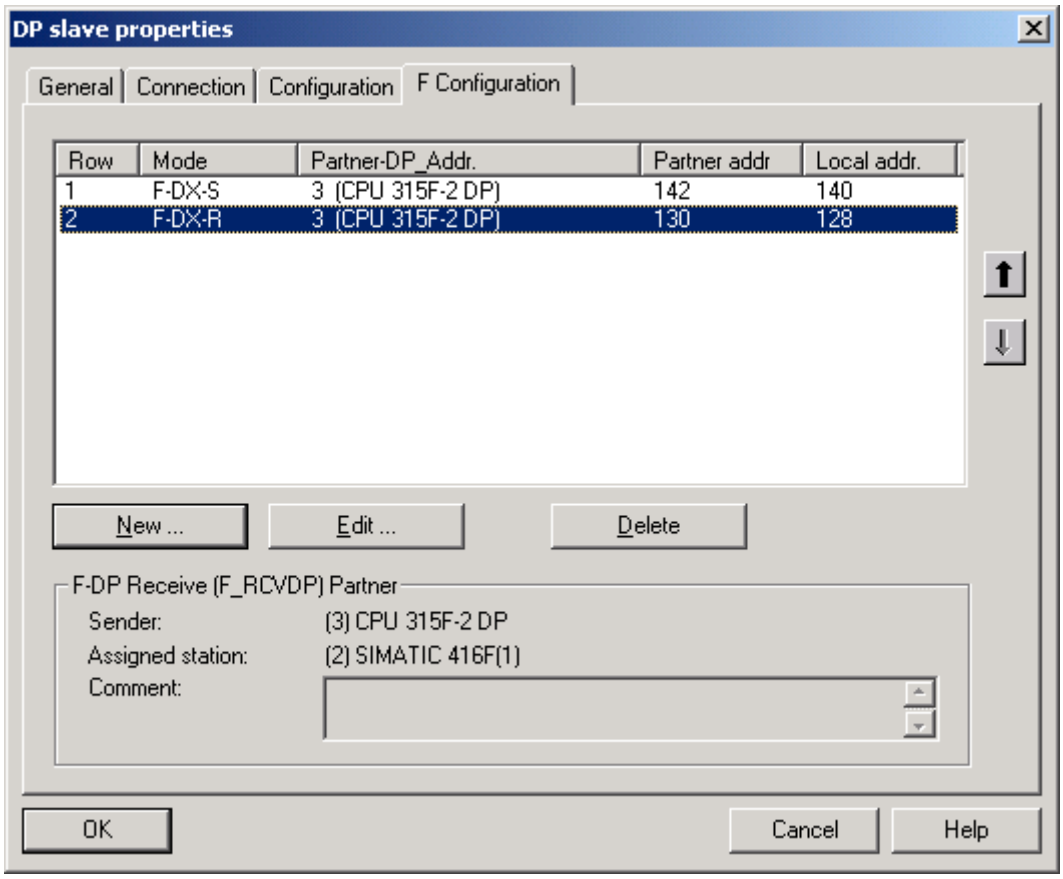
You have created a project in *STEP 7*.

1. Create a station in your project (in *SIMATIC Manager,* for example, an S7-300 station).
2. Assign a CPU with fail-safe capability to this station (in *HW Config*, from the hardware catalog).
3. Configure this CPU as a DP slave (in *HW Config*, in the "Operating Mode" tab of the Object Properties for the DP interface of the CPU).
4. After steps 1 to 3, configure a further DP slave (I-slave).
5. Create another station and assign a CPU with fail-safe capability (see steps 1 and 2).
6. Configure this CPU as a DP master (in *HW Config*, in the "Operating Mode" tab of the Object Properties for the DP interface of the CPU).
   Note: The CPU of the DP master can be an F-CPU or a standard CPU.
7. In the hardware catalog, under "Configured stations," select the station type of one I-slave (for example, the "CPU 31x") and place it on the DP master system.
8. Link the I-slave to the DP master in the Connection dialog box that opens automatically.
9. After steps 7 and 8, link the second I-slave to the DP master.
   You can now specify the address areas for the safety-related I-slave to I-slave communication:
10. In the "F-Configuration" tab of the Object Properties of I-slave 1, select "New."
11. In the next dialog, make the following entries for the receive connection from I-slave 2 for our example:
    - For "Mode: F-MS-R" (receive over fail-safe I-slave to I-slave communication)
    - for "DP partner (sender): DP address: 2: Slave; address (LADDR): 130"
    - For "local (receiver): address (LADDR): 128"
    - Accept the defaults for the other parameters in the dialog.

The dialog appears as shown below:



12. Confirm your entries with "OK."
13. In the "F-Configuration" tab of the Object Properties of I-slave 1, select "New."
14. In the next dialog, make the following entries for the send connection to I-slave 2 for our example:
    - For "Mode: F-DX-S" (send over fail-safe I-slave to I-slave communication)
    - for "DP partner (receiver): DP address: 2: Slave; address (LADDR): 142"
    - For "local (sender): address (LADDR): 140"
    - Accept the defaults for the other parameters in the dialog.
15. Confirm your entries with "OK."
    This results in two configuration rows for this example:



Note

In the object properties of the relevant I-slave, entries are automatically made in the "Configuration" tab based on the configuration in the "F-Configuration" tab. These entries must not be modified. Otherwise, safety-related I-slave to I-slave communication is not possible.

The assigned address areas on the DP master and I-slaves can be seen in the "Configuration" tab.

Additional Information

You will find a description of the parameters in the *context-sensitive online help of the "F-Configuration" tab*.

For information on programming safety-related I-slave to I-slave communication, refer to Section *1.2.6*.

For information on the address areas, process image partitions, and supported interrupt OBs, refer to the Technical Specifications of the CPU you are using.

1.1.5 Configuring the F-CPU

Information on Local Data

The information below on defining local data for the safety program **completely replaces** the corresponding information in Section *3.3* of the manual.

"F-Local Data" Parameter

F-blocks are automatically added when the safety program is compiled to create an executable safety program from your safety program. Use this parameter to define the amount of local data in bytes that is available for the automatically added F-blocks and the F-CALL.

Note

You must provide **at least 300 bytes** of local data for the safety program. However, the local data requirement for the automatically added F-blocks may be higher depending on the requirements of your safety program.

Thus, you should provide as much local data as possible for the automatically added F-blocks.If the amount of local data available for the automatically added F-blocks is insufficient (less than 300 bytes), the runtime of the safety program increases. You will receive a notice via *S7 Distributed Safety*, if the automatically added F-blocks would require more local data than configured. The safety-related program will be generated anyway.

Safety Note

The calculated maximum runtime of the safety program using the MS Excel file in the directory (*...\Siemens\STEP7\S7Manual\s7fco\s7fcotib.xls)* is no longer be correct in this case, because the calculation assumes sufficient F-local data are available.

In this case, use the value you configured for the max. cycle time of the F run-time group (F-monitoring time)to calculate the max. reaction times in the event of an error and for any run-time of the standard system using the above-mentioned Excel file.

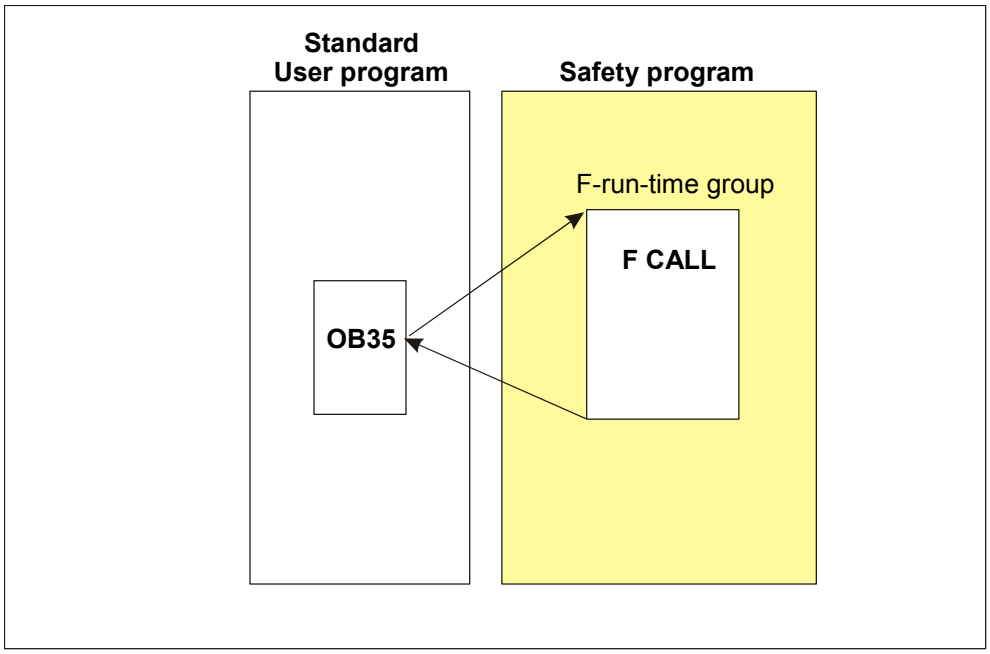
Note

Note that the maximum possible amount of F-local data depends on the following:

- Local data requirement of your higher-level standard user program
  For this reason, you should call the F-CALL block directly in the OB (cyclic interrupt OB35, if possible) and not declare any additional local data in the cyclic interrupt OB.
- Maximum amount of local data of the utilized F-CPU (see *Technical Specifications in the Product Information*)
  For CPU 416F-2, you can configure the local data for each priority class. For this reason, you should allocate the largest possible area for local data for the priority class in which the safety program is called (for example, OB35).

Maximum Possible Amount of F-Local Data According to Local Data Requirement of Higher-Level Standard User Program

Case 1: F-CALL called directly in the OB

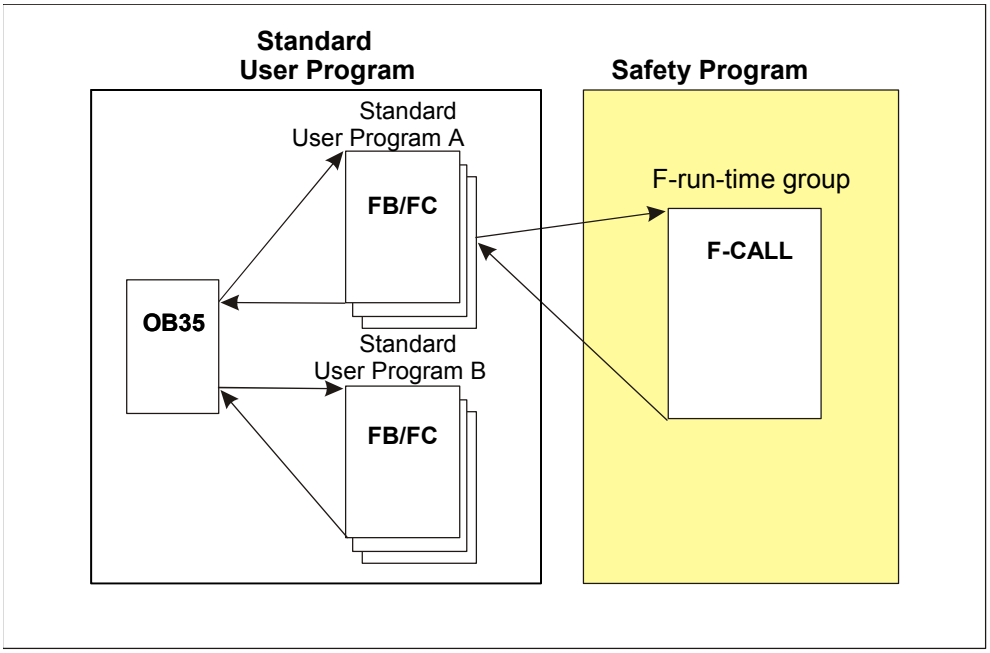


Set the "F-local data" parameter to the following:

- The maximum size of the local data of the F-CPU you are using minus 32 bytes or
- the maximum size of the local data of the F-CPU you are using minus local data requirements of the OB, if this is greater than 32 bytes.

Comment: You can derive the local data requirement of the OB from the program structure. In *SIMATIC Manager*, select the **Options > Reference Data > Display** menu command (Setting: "Program structure" selected). This shows you the local data requirement in the path or for the individual blocks (see also *STEP 7 Help*).

Case 2: F-CALL not called directly in the OB



Set the "F-local data" parameter to the following:

- The maximum size of the local data of the F-CPU you are using minus 32 bytes or
- the maximum size of the local data of the F-CPU you are using minus local data requirements of the OB and minus local data requirements of the standard user program A, if these are more than 32 bytes together.

Comment: You can derive the local data requirement of the OB or the standard user program A from the program structure. In *SIMATIC Manager*, select the **Options > Reference Data > Display** menu command (Setting: "Program structure" selected). This shows you the local data requirement in the path or for the individual blocks (see also *STEP 7 Help*).

Local Data Requirement for the Automatically Added F-Blocks According to Local Data Requirement of User Safety Program

The information below must be taken into account only if the amount of local data available for your safety program is insufficient and you received a message from *S7 Distributed Safety* to that effect.

You can estimate the probable local data requirement for the automatically added F-blocks as follows:

Determine the local data requirement for each call hierarchy (path starting from the F-PB across all nesting levels down to the lowest) of your safety program:

Local data requirement for a call hierarchy (path local data requirement in bytes) =

2 x amount of all local data of F-FBs/F-FCs of data type BOOL in the path

\+ 4 x amount of all local data of F-FBs/F-FCs of data type INT in the path

\+ 6 x amount of all local data of F-FBs/F-FCs of data type TIME in the path

\+ 22 x number of nesting levels in which an F-application block is called

\+ 42 x number of nesting levels

\+ 14 x number of nesting levels in which a fixed-point function is programmed

The estimated local data requirement for the automatically added F-blocks is thus equivalent to the maximum of the path local data requirement of all paths.

Note

If you are unable to provide a sufficient amount of local data for the automatically added F-blocks, we recommend that you reduce the local data requirement of your safety program, by reducing nesting depth, for example.

Use of Local Data in an F-FB or F-FC

Note

F-blocks are automatically added when the safety program is compiled to create an executable safety program from your safety program. If you use the local data memory area in an F-FB/F-FC, remember the following limit (irrelevant for F-CPUs from the S7-400 range):

Local data requirement< max. size of local data per block
(see *Technical Specification* about the F-CPU used in the Product Information

Mean local data requirement in bytes =

2 x amount of local data of F-FB/F-FC of data type BOOL

+ 4 x amount of local data of F-FB/F-FC of data type INT

+ 6 x amount of local data of F-FB/F-FC of data type TIME

+ 12

+ 14 (if a fixed-point function is programmed)

+ 6 (if an F-FB, F-FC, or F-application block is called)

+ 24 (if a called F-FB, F-FC or F-application block contains a parameter of data type TIME)

If the amount of local data required is greater, you cannot download your safety program to the F-CPU. Reduce the local data requirement of your programmed F-FB or F-FC.

1.2 Programming the Safety Program

1.2.1 Differences between F-Programming Languages and Standard Programming Languages

Access to Data Blocks

Data blocks should always be accessed with "fully qualified DB access" to ensure that the correct data block is opened.

The initial access to data of a data block in an F-FB/F-FC **must** always be a "fully qualified DB access," or it must be preceded by the "OPN DB" operation. This also applies to the initial access to data of a data block after a jump label.

Options for the data blocks "Unlinked" and "DB is write-protected in the AS"

Note

The adjustable option "Unlinked" in the object properties of a DB must not be set for the F-DBs and instance DBs of the F-blocks.

The selectable option "DB is write-protected in the AS" in the object properties of the of a DB must not be set for F-DBs and instance DBs of F-blocks.

If you have set one of the options listed above, this is corrected when you compile the safety program.

Boolean Constants "0" and "1"

If you require the Boolean constants "0" and "1" in your safety program to assign values to parameters during block calls, you can use the tags "RLO0" and "RLO1" in the F shared DB with full qualified DB access ("F_GLOBDB".RLO0 or "F_GLOBDB".RLO1).

1.2.2 FBD/LAD Operations

New Operations Supported

You can use the operations listed in the table below in the safety program.

Operation		Function	Description
F-FBD	**F-LAD**		
JMP	--(JMP)	Jump operation	Unconditional jump in block Jump in block if 1 (conditional)
JMPN	--(JMPN)	Jump operation	Jump in block if 0 (conditional)
RET	--(RET)	Programmed control	Return (exit block)
OV	OV --| |--	Status bit	Evaluate exception bit overflow (OV bit in status word)

Note

- An F_SENDDP call must not be programmed between a jump operation and its associated destination.
- For F-PB only: A RET operation must not be programmed before a F_SENDDP call.

Safety Note

If you call the following F application blocks in your safety program: F_TP, F_TON, F_TOF, F_ACK_OP, F_2HAND, F_MUTING or F_1oo2DI and use the operations JMP, JMPN, or RET, remember the following:

- Each call for the F application blocks listed must only be processed once in a cycle of the F run-time group. In other words, the calls for the listed F application blocks must not be processed more than once by the JMP or JMPN operations (loop).
- As soon as a timer (PT, timer in F_ACK_OP, DISCTIME, DISCTIM1/2 or TIME_MAX) starts and has not yet elapsed, a call for one of the listed F application blocks must be processed in every cycle of the F run-time group. In other words, the calls for the listed F application blocks must the not be skipped by the JMP, JMPN, or RET operations (branch).

It is therefore advisable, to process the calls of the listed F application blocks exactly once in every cycle of the F run-time group because the timers cannot be correctly updated if this restriction is ignored.

Please note how the F application blocks work and refer to the timing diagrams (see manual, Section *5.7.3*).

OV Bit Evaluation

By evaluating the OV bit, you can identify an overflow without the F-CPU going into STOP mode in the case of an overflow.
(If you do not evaluate the OV bit, an overflow causes the F-CPU to switch to STOP mode if the result/quotient in an output is fed to an F I/O, or to a partner F-CPU by means of safety-related CPU-to-CPU communication.)

If you want to program an OV bit scan, observe the following conditions:

Note

An OV bit scan is only permitted in the network following the network with the operation that affected the OV bit.

The network with the OV bit scan must not be the destination of jump operation; in other words, it must not contain a jump label.

If an OV bit scan is programmed in the network following the operation that influences the OV bit, the execution time of the operation influencing the OV bit is prolonged (see also *Excel file for calculating the response time* in the *...\Siemens\STEP7\S7Manual\s7fco\s7fcotib.xls folder)*.

Safety Note

If an OV bit scan is programmed in the network following the operation that influences the OV bit and the result of the operation influencing the OV bit (an ADD_I-, SUB_I-, MUL_I or NEG_I operation or the quotient of a DIV_I operation) is outside the permitted range for integers (16 bits), the F-CPU does **not** change to STOP.

The result/quotient behaves like the analogous operation in a standard user program.

1.2.3 F I/O DB

Default Settings for QBAD and PASS_OUT

Contrary to the description in Section *5.3.2* of the manual, the default setting for the tags QBAD and PASS_OUT of the F-I/O DB is "1".

The default setting of "1" has no effect on safety programs that were created with *S7 Distributed Safety* V 5.1.

1.2.4 Using Substitute Values

SM 336; AI 6 x 13-Bit: Substitute Value Output

The F-system identifies an overflow or underflow of a channel of the SM 336; AI 6 x 13-bit as an F-I/O fault or channel fault. The fail-safe value 0 is provided in place of $7FFF_H$ (for overflow) or 8000_H (for underflow) in the PII for the safety program.

Use of Individual Substitute Values

If in the case of an F-I/O with inputs, you want to process other substitute values besides "0" in the safety program when substitute values are output, you can specify individual substitute values when QBAD = 1 (see manual, Section *5.3.2*).

1.2.5 Implementing a User Acknowledgment

This section supplements Section *5.3.9* of the manual.

1.2.5.1 Implementing a User Acknowledgment in the Safety Program of the F-CPU of a DP Master

See manual, Section *5.3.9.*

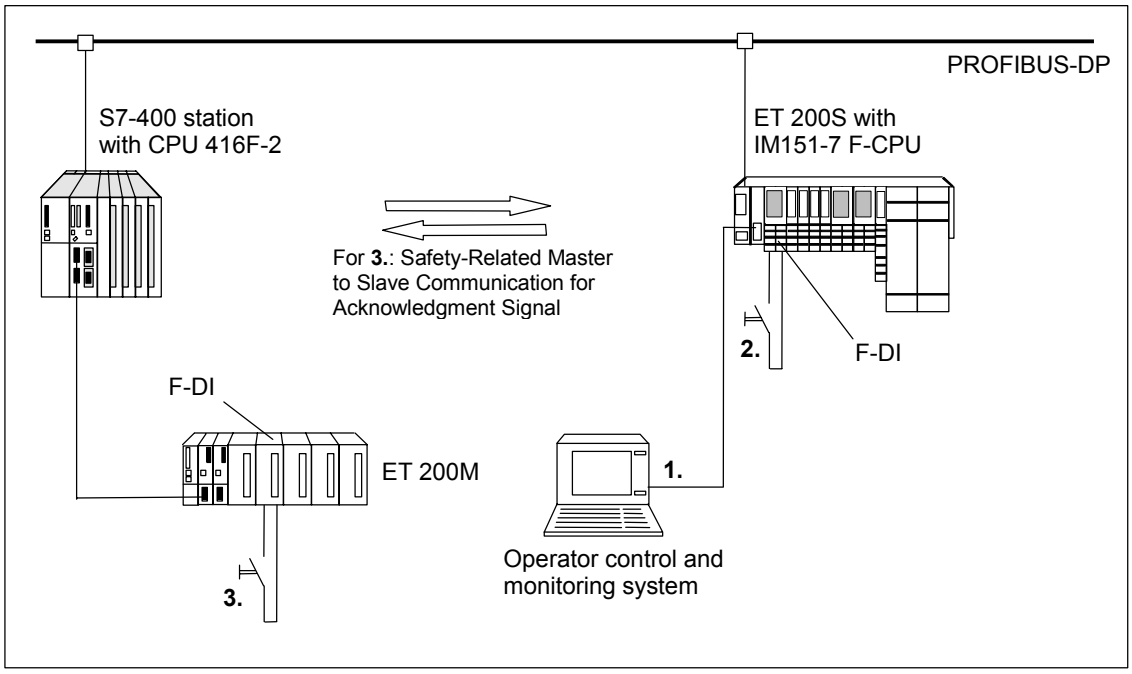
1.2.5.2 Implementing a User Acknowledgment in the Safety Program of the F-CPU of an Intelligent DP Slave

Options for User Acknowledgment

You can implement a user acknowledgment in one of the following ways:

1. An operator control and monitoring system with which you can access the F-CPU of the intelligent DP slave
2. An acknowledgment key that you connect at an F-I/O with inputs that is assigned to the F-CPU of the intelligent DP slave
3. An acknowledgment key that you connect at an F-I/O with inputs that is assigned to the F-CPU of the DP master

These three options are illustrated in the figure below.



1. User acknowledgment using an operator control and monitoring system with which you can access the F-CPU of the I-slave

To implement a user acknowledgment using an operator control and monitoring system, you require the F application block F_ACK_OP from the F-library *Distributed Safety* (V1) (see manual, Section *5.7.3.7*).

How to program the user acknowledgment using an operator control and monitoring system with which you can access the F-CPU of the I-slave

Follow the steps as described in the manual, Section *5.3.9* under *Procedure for Programming User Acknowledgment by Means of an Operator Control and Monitoring System*.

Call the F_ACK_OP F application block in the safety program of your I-slave. In so doing, note that the acknowledgment signal for evaluating user acknowledgments is provided at output **OUT** of F_ACK_OP ("Output Q" specified in the manual is incorrect!).

From your operator control and monitoring system, you can then access the instance DB of F_ACK_OP on the I-slave directly.

Also note the associated safety notes in the manual.

2. User acknowledgment with an acknowledgment button on an F-I/O with inputs that is assigned to the F-CPU of the intelligent DP slave

Note

If a communication error/F-I/O fault, or channel fault occurs in the F-I/O to which the acknowledgment button is connected, it is no longer possible to send an acknowledgment to reinclude this F-I/O.

This "block" can only be removed by a STOP/RUN change on the F-CPU of the intelligent DP slave .

To allow acknowledgment and reinclusion of an F-I/O to which an acknowledgment button is connected, it is advisable to plan a further acknowledgment over an operator control and monitoring system with which you can access the F-CPU of the I-slave (see 1.).

3. User acknowledgment using an acknowledgment button on an F-I/O with inputs that is assigned to the F-CPU of the DP master

If you also want to use the acknowledgment button assigned to the F-CPU on the DP master for a user acknowledgment in the safety program of the F-CPU of an intelligent DP slave, you must transfer the acknowledgment signal using safety-related master to I-slave communication (see Section *1.1.2* and *1.2.6*) from the safety program in the F-CPU of the DP master to the safety program in the F-CPU of the intelligent DP slave.

Procedure for Programming User Acknowledgment by Means of an Acknowledgment Key at an F-I/O with Inputs that is Assigned to the F-CPU of the DP Master

1. Call the F-application block F_SENDDP in the safety program in the F-CPU of the DP master
   (see manual, Section *5.4*).
2. Call the F-application block F_RCVDP in the safety program in the F-CPU of the intelligent DP slave (see manual, Section *5.4*).
3. Supply an input SD_BO_xx of the F_SENDDP block with the input of the acknowledgment key.
4. The acknowledgment signal for evaluating user acknowledgments is now available at the corresponding output RD_BO_xx of the F_RCVDP block. The acknowledgment signal can now be read in the program sections in which further processing is to take place with fully qualified access directly in the associated instance DB (for example, "Name F_RCVDP1".RD_BO_02). To enable this, you must first assign a symbolic name (Name F_RCVDP1" in the example) for the instance DB of F_RCVDP in the symbol table.
5. Supply the corresponding input SUBBO_xx of the F_RCVDP block with the fail-safe value "RLO0," so that an unintended user acknowledgment is not triggered before communication is established the first time after startup of the sending and receiving F-system, or in the event of a safety-related communication error. RLO 0 is provided in the F-Shared DB. At input SUBBO_xx, enter "F_GLOBDB".RLO0 fully qualified.

Note

If a communication error/F-I/O fault, or channel fault occurs in the F-I/O to which the acknowledgment button is connected, it is no longer possible to acknowledge to reinclude this F-I/O.

This "block" can only be removed by a STOP/RUN change of the F-CPU of the DP master.

To allow acknowledgment and reinclusion of an F-I/O to which an acknowledgment button is connected, it is advisable to plan a further acknowledgment over an operator control and monitoring system with which you can access the F-CPU of the DP master (see manual, Section *5.3.9*).

If a safety-related master to I-slave communication error occurs, the acknowledgment signal cannot be transferred, and an acknowledgment for reinclusion in the safety-related communication is no longer possible.

This "block" can only be removed by a STOP/RUN transition of the F-CPU of the intelligent DP slave.

To allow acknowledgment and reinclusion in safety-related communication, it is therefore advisable to plan a further acknowledgment over an operator control and monitoring system, with which you can access the F-CPU of the I-slave and transfer an acknowledgment signal (see 1.).

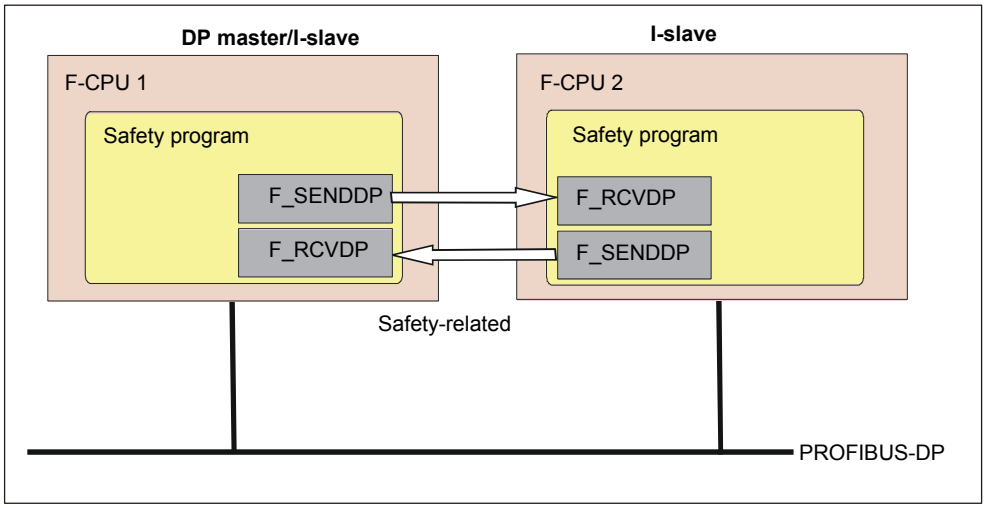
1.2.6 Programming Safety-related Master to I-Slave Communication and I-Slave to I-Slave Communication

Overview

The manual describes safety-related master to master communication.

The procedure for programming safety-related master to I-slave communication or safety-related I-slave to I-slave communication is exactly the same as for programming safety-related master to master communication. For this reason, only the differences are described in the following section.

Communication by Means of F_SENDDP and F_RCVDP



For safety-related communication between the F-CPUs of the DP master and an I-slave or between the F-CPUs of several I-slaves, you use the F application blocks F_SENDDP for sending and F_RCVDP for receiving. They can be used to transfer safely a *fixed* amount of fail-safe data of the data types BOOL and INT.

These library blocks are know-how protected. You will find them in the *F-Application Blocks* container in the *Distributed Safety* F-library. The F_RCVDP **must** be called at the start of the F-PB, and the F_SENDDP at the end of the F-PB.

A detailed description of the F-application blocks F_SENDDP and F_RCVDP can be found in the manual, in Section *5.7.3.10*.

How to Assign F-CPUs to F_SENDDP/F_RCVDP

Assign the F-CPUs to F_SENDDPs/F_RCVDPs as follows:

- Configure the address areas (local and partner addresses) for the DP master and the I-slave(s) in *HW Config* (see Sections *1.1.1*, *1.1.3* and *1.1.4*)
- Specify the following for master to I-slave communication in the safety program of the F-CPU of the DP master:
  - in F_SENDDP for input parameter LADDR, the partner address for sending ("F-Configuration" tab: Row Mode: "F-MS-S")
  - in F_RCVDP for input parameter LADDR, the partner address for receiving ("F-Configuration" tab: Row Mode: "F-MS-R")
- Specify the following for master to I-slave or I-slave to I-slave communication in the safety program of the F-CPU of an I-slave:
  - in F_SENDDP for input parameter LADDR, the local address for sending ("F-Configuration" tab: Row Mode: "F-MS-S" or "F-DX-S")
  - in F_RCVDP for input parameter LADDR, the local address for receiving ("F-Configuration" tab: Row Mode: "F-MS-R" or "F-DX-R")

Make these assignments for each F-CPU involved.

Note

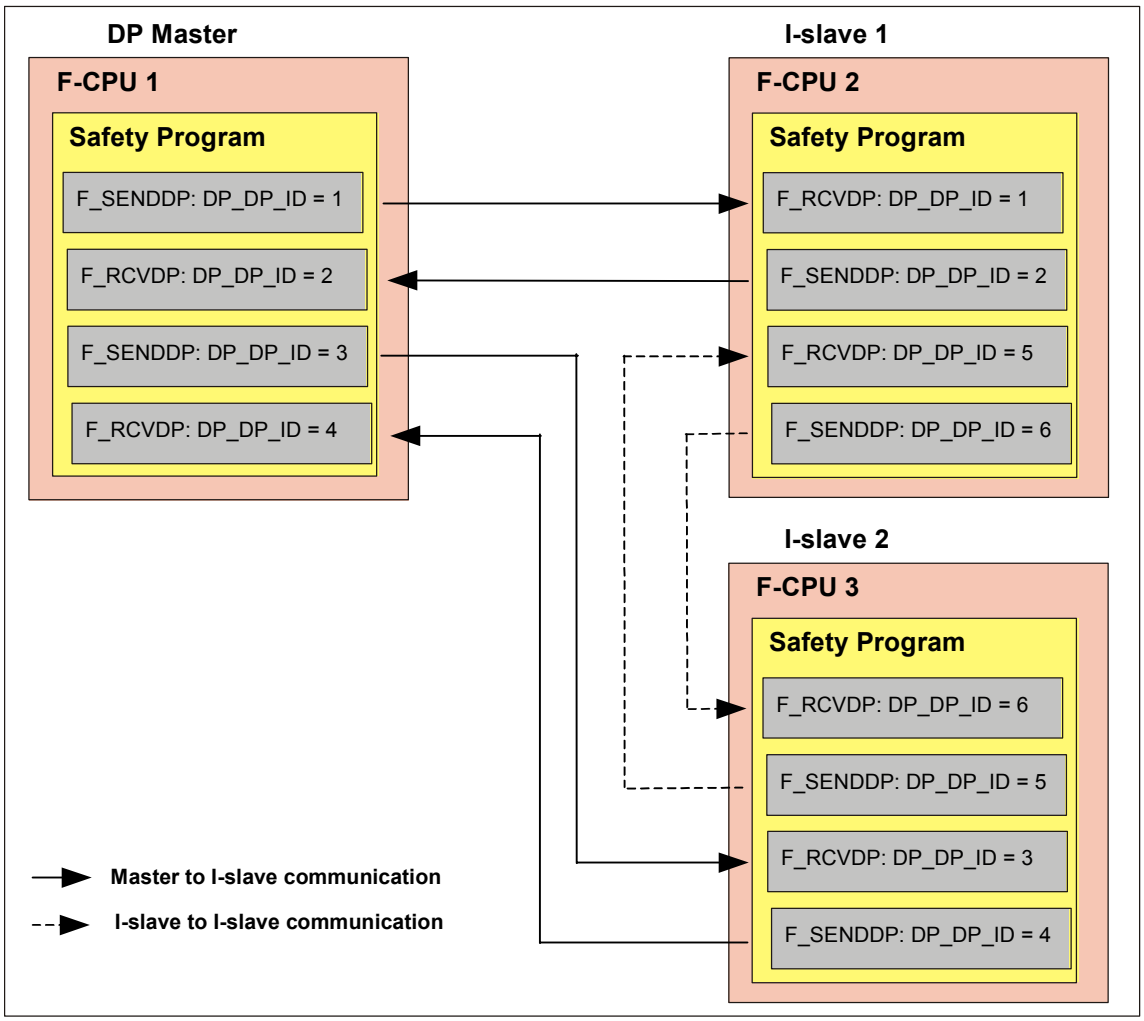
The settings for safety-related master to I-slave and I-slave to I-slave communication are always as follows:

- For F_SENDDP/F_RCVDP of the **DP master** always enter **the partner addresses** for the communication connections (from *HW Config*, "F-Communication" tab of the I-slave).
- For F_SENDDP/F_RCVDP of a **DP slave** always enter **the local addresses** for the communication connections (from *HW Config*, "F-Communication" of the I-slave).

Programming Procedure

The procedure for programming safety-related master to I-slave communication or I-slave to I-slave communication is exactly the same as for programming safety-related master to master communication. Refer to Section *5.4* of the manual.

In the schematic below, the example from the manual is simply adapted to specify the address relationships at the inputs of the F application blocks F_SENDDP and F_RCVDP for two safety-related master to I-slave and one I-slave to I-slave communication relations.



Safety Note

The value for each address association (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network.

Note

A separate instance DP must be used for each call of an F SENDDP or F_RCVDP block.

Limits for Data Transfer

If the amount of data to be transmitted is greater than the capacity of a F_SENDDP/F_RCVDP block pair, you can use additional F_SENDDP/ F_RCVDP block pairs. These require further communication connections. Remember the maximum limit of 244 bytes of input and 244 bytes of output data for transfer between an I-slave and a DP master.

The following table shows you how many output and input data are required for safety-related communication connections:

Safety-related Communication	Communication Connection	Required Input and Output Data			
		Between I-Slave 1 and DP Master		Between I-Slave 2 and DP Master	
		Output Data	Input Data	Output Data	Input Data
Master to I-slave	Send: I-slave 1 to DP master	12 bytes	6 bytes	-	-
	Receive: I-slave 1 from DP master	6 bytes	12 bytes	-	-
I-slave to I-slave	Send: I-slave 1 to I-slave 2	12 bytes	-	6 bytes	-
	Receive: I-slave 1 from I-slave 2	6 bytes	-	12 bytes	-

Within the maximum limit of 244 bytes of input and 244 bytes of output data for transfer between an I-slave and a DP master, remember any master to slave connections (MS) or direct data exchange connections (DX), on which you exchange data within your standard user program.

You can check whether you are within the maximum limit of 244 bytes of input and 244 bytes of output data for all the configured safety-related and standard communication connections in the "Configuration" tab in the object properties of the I-slave. Include all rows with MODE "MS" in the "Configuration" tab. The rows with MODE "DX" are not included.

User Acknowledgment

To implement a user acknowledgment in the safety program of the F-CPU of an intelligent DP slave, refer to the description in Section 1.2.5.

1.2.7 F-Shared DB

F-Shared DB

In addition to the items specified in Section *5.5* of the manual, you can read out the following in the standard user program or by means of an operator control and monitoring system:

- Compile data of the safety program (tag "F_PROG_DAT", data type DATE_AND_TIME)

Note

Starting with *S7 Distributed Safety* V5.2, the collective signature of the safety program (tag "F_PROG_SIG") is output in the F-Shared DB as a double word.

If you used *S7 Distributed Safety* V 5.1 previously and read out the tag "F_PROG_SIG" in the safety program or by means of an operator control and monitoring system, and you are now converting to *S7 Distributed Safety* V 5.2 + SP 1, you must change the data type to DWORD for evaluation.

The following can be read out in the safety program in the F-Shared DB:

- Constants 0 and 1 (tags "RLO0" and "RLO1", data type BOOL)

These tags are accessed fully qualified (for example, "F_GLOBDB".RLO0). The number and symbolic name of the F-shared DB and the absolute address of the tags are indicated in the printout of the safety program.

1.2.8 Creating F-Blocks in F-FBD/F-LAD

"Check Block Consistency" Function

The "Check block consistency" function can be found in *SIMATIC Manager* in the "Edit" menu, if you have selected a block container.

The "Check block consistency" function rectifies many of the time stamp conflicts and block inconsistencies. You can use this function in your safety program for F-FBs, F-FCs, and F-DBs. The procedure is the same as for a standard system

The "Check block consistency" function is not sufficient for obtaining a consistent safety program. Rather, you must compile the safety program (see manual, Section *5.8.3*).

1.2.9 Know-how Protection for F-FBs and F-FCs Written by the User

Know-how Protection

A block with know-how protection is a protected block that cannot be edited.

As of *S7 Distributed Safety*, V 5.2 + SP 1, you can assign know-how protection to F-FBs and F-FCs you yourself have created.

The protected F-FBs/F-FCs can no longer be modified.

You can only view the block properties and the tag declaration section of protected F-FBs/F-FCs; the statement section remains hidden.

The tag declaration table of the F-FBs/F-FCs display the same tag declaration types as the standard blocks with know-how protection.

Using Know-how Protection

Use know-how protection when you want to protect the knowledge contained in an F-FB/F-FC and want to prevent unwanted manipulation of the F-FBs and F-FCs.

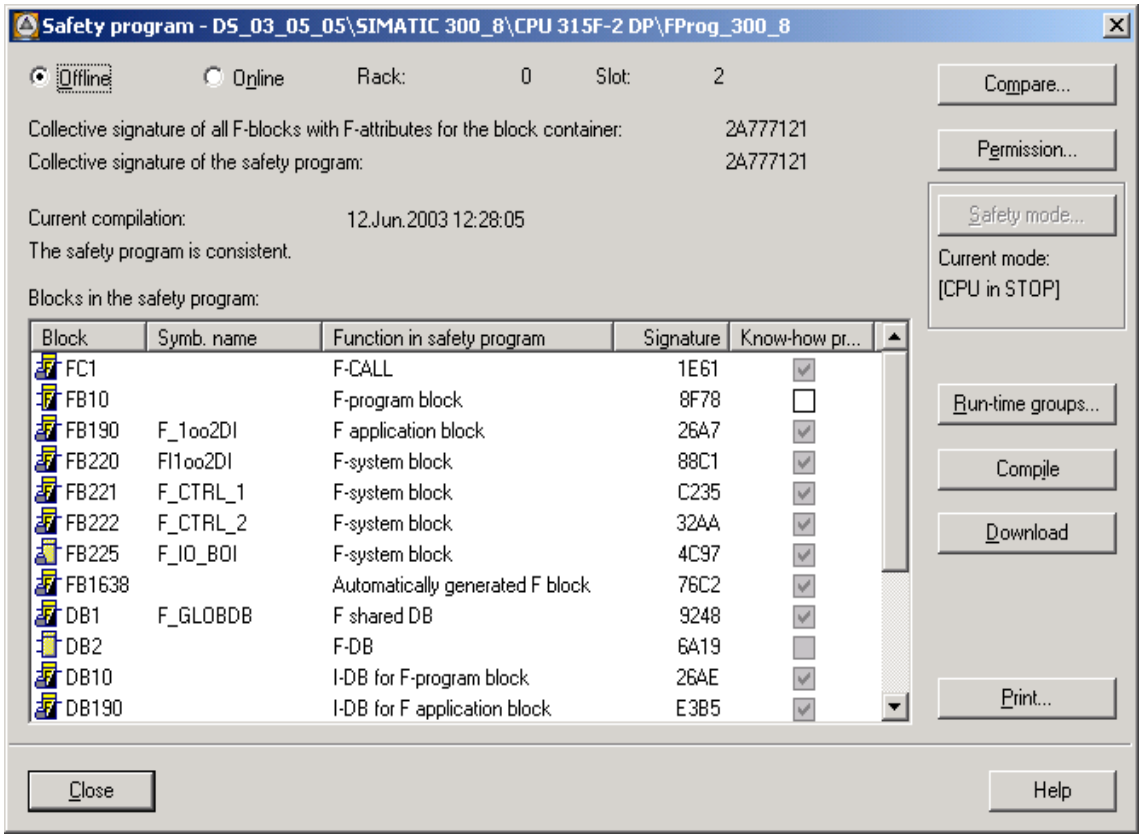
Setting Know-how Protection

Requirements:

You have created F-FBs or F-FCs and you want to protect the know-how they contain. The F-FBs/F-FCs you want to protect are not open in the *FBD/LAD Editor*.

Follow the steps outlined below:

1. Open the "Safety Program" dialog in the *SIMATIC Manager*.
2. You set know-how protection for F-FBs/F-FCs in the off-line safety program. So select "Offline".



3. Check the relevant check box in the "Know-how protection" column for the F-FBs and F-FCs.
   Result: A dialog for creating a backup copy opens automatically for every F-FB/F-FC you want to protect.
4. Remember the following when you save the backup copy:

Note

Assign a unique name for the backup copy so that you can later identify the F-FB/F-FC and the protected F-FB/F-FC (for example, same name, comment on F-FB/F-FC).

Do not save the backup copy in the project containing the protected F-FB/F-FC (otherwise an unprotected copy of the F-FB/F-FC is available).

If you want to save the backup copy in a library, make sure that it is a user-created F-library *in S7 Distributed Safety* (see Section *1.2.14*). The *FBD/LAD Editor* displays only F libraries for *S7 Distributed Safety*.

5. Save the backup copy of the F-FB/F-FC.
   Result: The check box in the "Know-how protection" column of the "Safety Program" dialog is activated and can no longer be selected.
   The block icon in the "Block" column has a padlock. The F-FB or F-FC is protected.
6. Follow the same procedure until all the F-FBs/F-FCs you want to protect are protected.

Modifying Protected F-FBs/F-FCs

Note

You cannot cancel the know-how protection of F-FBs/F-FCs.

If you want to modify a protected F-FB/F-FC, follow the steps below:

1. Delete the protected F-FB/F-FC from your project.
2. Copy the backup copy of the F-FB/F-FC into your project.
3. Edit the unprotected F-FB/F-FC in the *FBD/LAD Editor*.
4. If required, set the know-how protection for the F-FB/F-FC (see above).

1.2.10 Distributed Safety F-Library (V1)

Introduction

This section supplements Section 5.7 of the manual and describes the changes to the *Distributed Safety* F-library (V1)
in *S7 Distributed Safety* V 5.2 + SP 1.

1.2.10.1 Changes

Changing F-Application Block Numbers

Note

Contrary to what is stated in the manual, you are permitted to change the F-application block numbers.

Note, too, that the symbolic name of an F-application block in the symbol table must match the name in the object properties of the block (header).

You cannot use symbolic names of F-application blocks of the F-*Library Distributed Safety* (V1) for user-created F-FBs, F-FCs, and blocks.

F_ACK_OP

The following note supplements the manual, Section 5.7.3.7.

Note

In the safety program, read access to the in/out IN in the corresponding instance DB are not permitted!

F_SENDDP and F_RCVDP

The following information supplements Section *5.7.3.10* of the manual.

The F-application blocks F_SENDDP and F_RCVDP are used as follows:

- for safety-related master to master communication
- for safety-related master to I-slave communication
- for safety-related I-slave to I-slave communication

The information in Section *5.7.3.10* of the manual also applies. In addition, note the modified description for the input LADDR given below. This description applies to F_SENDDP and F_RCVDP:

	Parameter	Data Type	Description	Default
Input:	LADDR	INT	Start address of the address area: • of the DP/DP coupler for safety-related master to master communication • in safety-related master to I-slave communication • in safety-related I-slave to I-slave communication	0

Note

In the safety program, read and write access to the inputs DP_DP_ID and LADDR in the corresponding instance DB are not permitted!

A separate start address must be configured at the LADDR input for each F_SENDDP and F_RCVDP call within the safety program.

1.2.11 FB 179 "F_SCA_I": Scaling Values of Data Type INT

Connectors

	Parameter	Data Type	Description	Default
Inputs	IN	INT	Input value to be scaled in physical units	0
	HI_LIM	INT	Upper limit value in physical units	0
	LO_LIM	INT	Lower limit value in physical units	0
Outputs	OUT	INT	Result of scaling	0
	OUT_HI	BOOL	1 = input value > 27,648: OUT = HI_LIM	0
	OUT_LO	BOOL	1 = input value < 0: OUT = LO_LIM	0

Mode of Operation

This F-application block scales the value at input IN in physical units between the lower limit value at input LO_LIM and the upper limit value at input HI_LIM. It is assumed that the value at input IN is between 0 and 27,648. The scaling result is provided at output OUT.

The F-application block operates according to the following equation:

OUT = [IN * (HI_LIM–LO_LIM)] / 27648 + LO_LIM

So long as the value at input IN is greater than 27,648, the output OUT is linked to HI_LIM, and OUT_HI is set to 1.

So long as the value at input IN is less than 0, the output OUT is linked to LO_LIM, and OUT_LO is set to 1.

For reverse scaling, you must assign LO_LIM > HI_LIM. With reverse scaling, the output value at output OUT decreases while the input value at input IN increases.

Performance in the Event of Overflow or Underflow of Analog Values and Fail-Safe Value Output

Note

If inputs from the PII of an SM 336; AI 6 x 13 bit are used as input values, you must bear in mind that the F-system detects an overflow or underflow of a channel of this F-SM as an F-I/O fault or channel fault. The fail-safe value 0 is provided in place of $7FFF_H$ (for overflow) or 8000_H (for underflow) in the PII for the safety program.

If other fail-safe values are to be output in this case, you must evaluate the QBAD tag in the F-I/O DB (branch to output of an individual fail-safe value).

If the value in the PII of the F-SM is within the overrange or underrange, but is greater than 27648 or less than 0, you can likewise branch to the output of an individual fail-safe value by evaluating the outputs OUT_HI or OUT_LO.

1.2.12 FC 178 "F_INT_WR": Writing a Value of the Data Type INT indirectly into an F-DB

Connectors

	Parameter	Data Type	Description
Inputs	IN	INT	Value to be written to the F-DB
	ADDR_INT	POINTER	Start address of the INT area in an F-DB
	END_INT	POINTER	End address of the INT area in an F-DB
	OFFS_INT	INT	Address offset in the INT area

Mode of Operation

This F application block writes the value applied to input IN of the data type INT to the tag in an F-DB addressed by ADDR_INT and OFFS_INT.

Via the ADDR_INT input, the start address of the area with tags of the data type INT is transferred to an F-DB in which the value is written to the IN input. The address offset in this area is transferred via the OFFS_INT input.

The addresses transferred at the ADDR_INT or END_INT inputs must point to a tag of the INT data type in an F-DB. There must only be tags of the INT data type between the addresses ADDR_INT and END_INT. The ADDR_INT address must be lower than the END_INT address. The transfer of the ADDR_INT and END_INT addresses must be fully qualified as "DBx.DBWy" or in the corresponding symbolic representation. Transfers in other forms are not permitted.

Diagnostic Result

The F application block checks whether the address of the tag addressed with ADDR_INT and OFFS_INT is outside the address area defined by the addresses ADDR_INT and END_INT. If this is the case, the F-CPU changes to STOP.

The following diagnostic event is then entered in the diagnostic buffer of the F-CPU:

Event ID 16#75E2
Safety program: area length error
occurred in: F application block F_INT_WR
word access to F-DB
F-DB number: x
access address: y
previous operating mode: RUN
requested operating mode: STOP
internal error, event entering state

Note the following: When using *STEP* 7, ≤ V 5.2 + SP 1, the diagnostic event is displayed only in hexadecimal without textual information. You then receive the F-DB number in hexadecimal as additional information 1, the access address in hexadecimal as additional information 3.

Examples of the Parameter Assignment of ADDR_INT, END_INT, and OFFS_INT

Address	Declaration	Name	Type	Initial Value	Comments
0.0	stat		STRUCT		
+0.0	stat	VAR_BOOL10	BOOL	FALSE	
+0.1	stat	VAR_BOOL11	BOOL	FALSE	
+0.2	stat	VAR_BOOL12	BOOL	FALSE	
+0.3	stat	VAR_BOOL13	BOOL	FALSE	
+2.0	stat	VAR_TIME10	TIME	T#0MS	
+6.0	stat	VAR_TIME11	TIME	T#0MS	
+10.0	stat	VAR_INT10	INT	0	<- ADDR_INT = "F-DB".VAR_INT10 Example 1
+12.0	stat	VAR_INT11	INT	0	
+14.0	stat	VAR_INT12	INT	0	
+16.0	stat	VAR_INT13	INT	0	<- OFFS_INT = 3
+18.0	stat	VAR_INT14	INT	0	
+20.0	stat	VAR_INT15	INT	0	<- END_INT = "F-DB".VAR_INT15
+22.0	stat	VAR_BOOL20	BOOL	FALSE	
+22.1	stat	VAR_BOOL21	BOOL	FALSE	
+22.2	stat	VAR_BOOL22	BOOL	FALSE	
+22.3	stat	VAR_BOOL23	BOOL	FALSE	
+24.0	stat	VAR_INT20	INT	0	<- ADDR_INT = "F-DB".VAR_INT20 <- OFFS_INT = 0 Example 2
+26.0	stat	VAR_INT21	INT	0	
+28.0	stat	VAR_INT22	INT	0	
+30.0	stat	VAR_INT23	INT	0	<- END_INT = "F-DB".VAR_INT23
+32.0	stat	VAR_INT30	INT	0	<- ADDR_INT = "F-DB".VAR_INT30 Example 3
+34.0	stat	VAR_INT31	INT	0	<- OFFS_INT = 1
+36.0	stat	VAR_INT32	INT	0	
+38.0	stat	VAR_INT33	INT	0	
+40.0	stat	VAR_INT34	INT	0	<- END_INT = "F-DB".VAR_INT34
+42.0	stat	VAR_TIME20	TIME	T#0MS	
=46.0	stat		END_STRUCT		

1.2.13 FC 179 "F_INT_RD": Reading a Value of the INT Data Type from an F-DB

Connectors

	Parameter	Data Type	Description
Inputs	ADDR_INT	POINTER	Start address of the INT area in an F-DB
	END_INT	POINTER	End address of the INT area in an F-DB
	OFFS_INT	INT	Address offset in the INT area
Outputs	OUT	INT	Value to be read from the F-DB

Mode of Operation

This F application block reads the tag addressed by ADDR_INT and OFFS_INT of the data type INT in an F-DB and applies it to the OUT output.

Via the ADDR_INT input, the start address of the area with tags of the data type INT in an F-DB from which the tag will be read is transferred. The address offset in this area is transferred via the OFFS_INT input.

The addresses transferred at the ADDR_INT or END_INT inputs must point to a tag of the INT data type in an F-DB. There must only be tags of the INT data type between the addresses ADDR_INT and END_INT. The ADDR_INT address must be lower than the END_INT address. The transfer of the ADDR_INT and END_INT addresses must be fully qualified as "DBx.DBWy" or in the corresponding symbolic representation as in the example in Section *1.2.12*. Transfers in other forms are not permitted.

You will find examples of the parameter assignment of ADDR_INT, END_INT and OFFS_INT in Section *1.2.12*.

Diagnostic Result

The F application block checks whether the address of the tag addressed with ADDR_INT and OFFS_INT is outside the address area defined by the addresses ADDR_INT and END_INT. If this is the case, the F-CPU changes to STOP.

The following diagnostic event is then entered in the diagnostic buffer of the F-CPU:

Event ID 16#75E2
Safety program: area length error
occurred in: F application block F_INT_RD
word access to F-DB
F-DB number: x
access address: y
previous operating mode: RUN
requested operating mode: STOP
internal error, event entering state

Note the following: When using *STEP* 7, ≤ V 5.2 + SP 1, the diagnostic event is displayed only in hexadecimal without textual information. You then receive the F-DB number in hexadecimal as additional information 1, the access address in hexadecimal as additional information 3.

1.2.14 FB 190 "F_1oo2DI": 1oo2 Evaluation with Discrepancy Analysis

Connectors

	Parameter	Data Type	Description	Default
Inputs	IN1	BOOL	Sensor 1	0
	IN2	BOOL	Sensor 2	0
	DISCTIME	TIME	Discrepancy time (0 ... 60 s)	T# 0 ms
	ACK_NEC	BOOL	1 = acknowledgment necessary for discrepancy error	1
	ACK	BOOL	Acknowledgment of discrepancy error	0
Outputs	Q	BOOL	Output	0
	ACK_REQ	BOOL	1 = acknowledgment necessary	0
	DISC_FLT	BOOL	1 = discrepancy error	0
	DIAG	Byte	Service information	0

Mode of Operation

This F application block implements a 1oo2 evaluation of two single channel sensors combined with a discrepancy analysis.

The output Q is set to 1, when the signal states of the two inputs IN1 and IN2 equal 1 and no discrepancy error DISC_FLT is stored. if the signal state of one or both inputs is 0, output Q is set to 0.

As soon as the signal states of the two inputs IN1 and IN2 are different, the discrepancy time DISCTIME is started. If the signal states of the two inputs are still different when the discrepancy time elapses, a discrepancy error is detected and DISC_FLT is set to 1 (restart disabled).

If no discrepancy is detected between inputs IN1 and IN2, the discrepancy error is acknowledged depending on the parameter assignment of ACK_NEC:

- If ACK_NEC = 0 the acknowledgment is automatic.
- If ACK_NEC = 1 you can only acknowledge the discrepancy error with a rising edge at input ACK.

The ACK_REQ = 1 output signals that a user acknowledgment is necessary at input ACK to acknowledge the discrepancy error (cancel the restart disable). The F application block sets ACK_REQ = 1 as soon as discrepancy is no longer detected. After acknowledgment or when there is once again a discrepancy between the inputs IN1 and IN2 before the acknowledgment, the F application block resets ACK_REQ to 0.

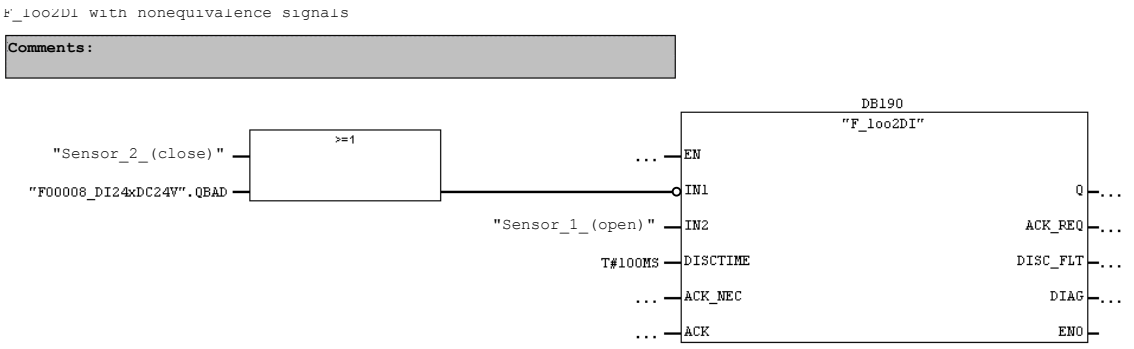
The output Q can never be set to 1 if the discrepancy time is set to values < 0 or > 60 s. In this case, the output DISC_FLT is also set to 1 (restart disabled). The call interval of the safety program (for example OB35) must be less than the set discrepancy time.

Activating inputs IN1 and IN2

The two inputs IN1 and IN2 must be activated so their positive state is 0.

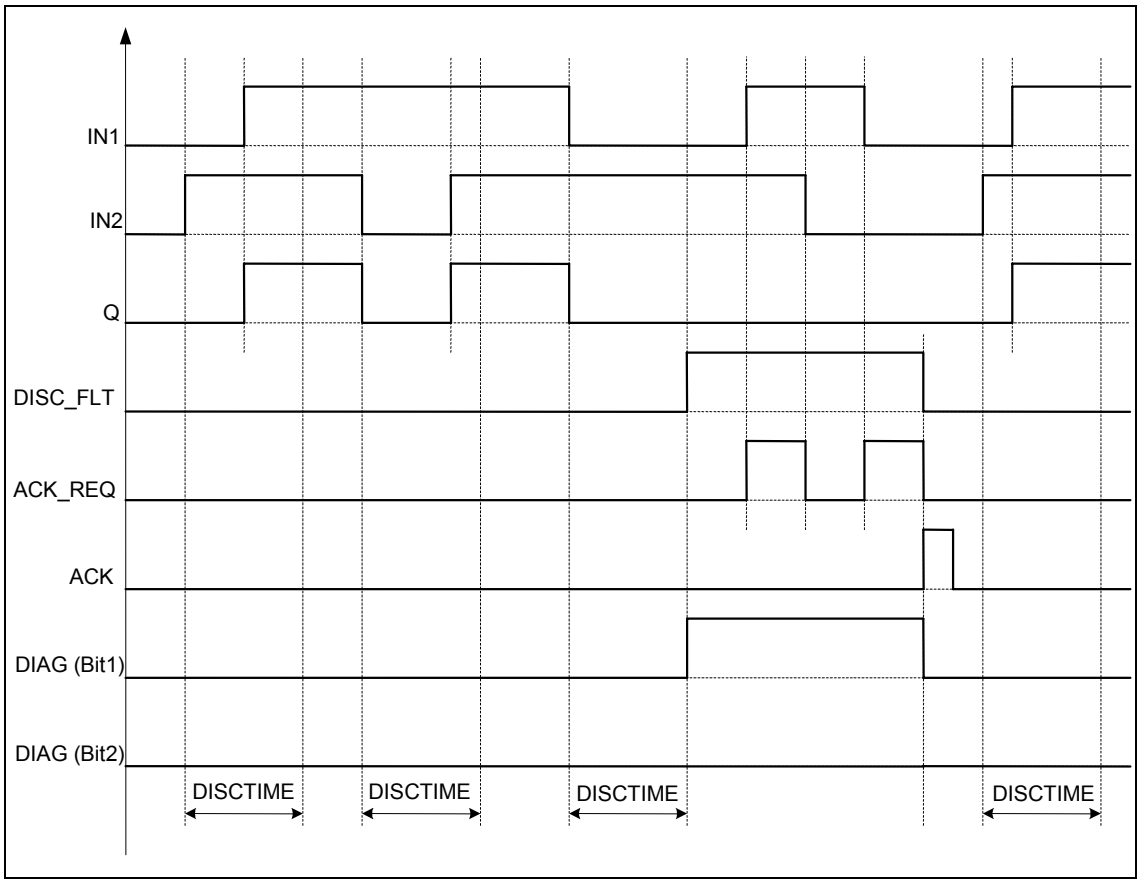
Example

For non-equivalent signals, you have to negate the input (IN1 or IN2), which you have assigned the positive state 1 for the sensor signal . You must also OR the sensor signal with the QBAD tags of the corresponding F-I/O DB, so that signal state 0 is applied to input IN1 or IN2 (after the negation), if substitute values are output.



Timing Diagram F_1oo2DI

If ACK_NEC = 1 is set:



Note

If the sensors at the inputs IN1 and IN2 are assigned to different F-I/Os, it is possible that the substitute values are output for different lengths of time following startup of the F system due to different startup response of the F-I/Os. If the signal states of the two inputs IN1 and IN2 remain different after the discrepancy time DISCTIME has elapsed, a discrepancy error is detected after the F system starts up.

If ACK_NEC = 1 you must acknowledge the discrepancy error with a rising edge at input ACK.

Safety Note

With times and periods of time you must take into account the inaccuracy that results from the cyclic processing. This is the same as the cycle time of your F run-time group.

You must also take into account the tolerance of the internal monitoring of the times in the F-CPU:

- for time values up to 100 ms maximum 20% of the set time value
- for time values higher than 100 ms maximum 2% of the set time value

DIAG Output

The DIAG output provides non fail-safe information on errors for service purposes. You can read this out using the operator control and monitoring system or evaluate it in your standard user program. The DIAG bits remain stored until you acknowledge at the ACK input.

Structure of DIAG

Bit no.	Meaning	Possible causes of problems	Remedies
Bit 0	Discrepancy error or bad discrepancy time set (= state of DISC_FLT)	Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		F-I/O error, channel error, or communication error*	Remedy see manual, Section *5.3.2, Bits 0 to 6 in Table "Structure of DIAG"*
		Discrepancy time set too low	Set higher discrepancy time
		Discrepancy time < 0 or > 60 s set	Set discrepancy time in range 0 through 60 s
Bit 1	If discrepancy error: Last signal state change was at input IN1	-	-
Bit 2	If discrepancy error: Last signal state change was at input IN2	-	-
Bit 3	Reserved	-	-
Bit 4	Reserved	-	-
Bit 5	If discrepancy error: Input ACK has permanent signal state 1	Acknowledgment button defective	Replace acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgment necessary (= state of ACK_REQ)	-	-
Bit 7	State of output Q	-	-

* If the sensors at inputs IN1 and IN2 are assigned to different F-I/Os, a discrepancy error can also occur because one of the F-I/Os outputs substitute values (QBAD in the relevant F-I/O DB = 1).

Note

Access to the DIAG output is not permitted in the safety program!

1.2.15 User-Created F-Libraries

User-Created F-Libraries

As of *S7 Distributed Safety*, V 5.2 + SP 1, you can create your own F-libraries for S7 Distributed Safety.

These "user-created F-libraries" can include only the following:

- F-FBs and F-FCs created by users in F-FBD/F-LAD
- Application templates created by users in F-FBD/F-LAD

Please note the following important information that differs from the information in the manual:

Note

In contrast to the information in the manual, in Section *5.6*, for *S7 Distributed Safety*, V 5.2 + SP 1 the supplied F-library *Distributed Safety* (V1) must not include user-created F-FBs, F-FCs, modified or additional application templates. You must create your own F-libraries for these objects.

The F-library *Distributed Safety* (V1) must only contain F-blocks and application templates that were installed with the *S7 Distributed Safety* version.

How to Create an F-Library

Requirements:

- You have created F-FBs, F-FCs in the *FBD/LAD editor* and possibly assigned know-how protection (see Section *1.2.9*), modified application templates of the F-library *Distributed Safety* (V1) or created new application templates.
- The F-FBs/F-FCs/application templates are not open in the *FBD/LAD editor*.

You create your own F-library with *S7 Distributed Safety*, V 5.2 + SP 1 as follows:

1. Open the F-library *Distributed Safety* (V1) and save it under another name (for example "PRESSCONTROL_V1").
2. Open the F-library "PRESSCONTROL_V1" and delete all its content (folder "F-Application Blocks" and "F-System Blocks" with their content).
3. Insert a new folder called "S7 Program" in the F-library "PRESSCONTROL_V1".
4. Open the project with the F-FBs/F-FCs/ application templates you have created and copy them to the "S7 Program" folder.
   You can rename the block container and further structure the F-library (create more "S7 Program" folders).

Note

if your new F-library is not displayed in the *FBD/LAD editor*, open and close the *FBD/LAD editor*.

By repeating the steps outlined above, you can create as many F-libraries for *S7 Distributed Safety* as you wish.

Working with User-Created F-Libraries

To use F-FBs/F-FCs/application templates from user-created F-libraries, you must have the *S7 Distributed Safety* version installed on your PC/PG with which the F-FBs, F-FCs or application templates were created.

You yourself must check whether or not an existing user-created F-library is still up to date. If necessary, you must replace a user-created F-library with a newer version. *S7 Distributed Safety* does not check the versions of the F-FBs/F-FCs in a user-created F-library. When you compile a safety program, there is also no automatic replacement of F-FBs/F-FCs from a user-created F-library with corresponding F-FBs/F-FCs from a newer version of this F-library. If necessary, copy the F-FBs/F-FCs with a new version from the user-created F-library into the block container of your current S7 program.

You cannot use symbolic names of F-application blocks of the F-*Library Distributed Safety* (V1) for user-created F-FBs, F-FCs, and blocks.

In terms of handling of the F-FBs/F-FCs/application templates from user-created F-libraries, there is no difference compared with the F-library *Distributed Safety* (V1).

Removing S7 Distributed Safety

When you remove *S7 Distributed Safety*, the user-created F-libraries are retained.

1.2.16 Compiling the Safety Program

Note

Before you compile the safety program, close the *LAD/STL/FBD editor*, *Display S7 Reference Data*, and *Check Block Consistency* applications.

1.2.17 Complete Function Test of Safety Program or Protection through Program Identification

Introduction

The following two sections replace the corresponding sections in the *Section 5.8.5* of the manual.

1.2.17.1 Transferring the Safety Program to the F-CPU with a Programming Device/PC

F-CPUs with an Inserted Memory Card (Flash Card or MMC)

The following safety notes applicable when the safety program is transferred from a programming device/PC to:

- F-CPUs with a flash card inserted (for example CPU 416F-2)
- F-CPUs with MMC
  (for example CPU 317F-2 DP, CPU 315F-2 DP or IM 151-7 F-CPU)

Safety Note

If the function of the safety program is not tested in the target F-CPU, you must comply with the following procedure when transferring the safety program to the F-CPU with a **programming device/PC** to ensure that the F-CPU does not contain an "old" safety program:

1. For F-CPUs with MMC: Download the safety program to the F-CPU in the "Safety Program" dialog.
   For F-CPUs with flash card inserted: Download the safety program to the F-CPU in the "Download User Program to Memory Card" dialog.
2. Perform a program identification (that is, check to determine whether the collective signatures of all F-blocks with an F-attribute in the block container match online and off-line; refer Sections *5.8.4* and *5.10.2* of the manual).
3. Perform a general reset of the F-CPU using the **mode selector** or by means of the **programming device/PC.** After deleting the work memory, the safety program is transferred from load memory (memory card, MMC on F-CPUs 3xxF and IM 151-7 F-CPU or flash card on F-CPUs 4xxF) to work memory.

Safety Note

If **more than one F-CPU** is accessible over a network (such as MPI) from **one programming device or PC**, you must take the following additional measures to ensure that the safety program is downloaded to the correct F-CPU:

Use passwords specific to each F-CPU, such as a uniform password for the F-CPUs having the respective MPI address as an extension: "Password_8".

Note the following:

- A point-to-point connection must be used the first time a password is assigned to an F-CPU (this also applies to the first time an MPI address is assigned to an F-CPU).
- Before downloading a safety program to an F-CPU that has not yet been assigned access protection with an F-CPU password, you must first revoke existing access permission for any other F-CPU.

F-CPUs without Flash Card

The following safety notes apply when the safety program is transferred from a programming device/PC to:

- F-CPUs without a flash card (for example CPU 416F-2)

Safety Note

If the function of the safety program is not tested in the target F-CPU, you must comply with the following procedure when transferring the safety program to the F-CPU with a **programming device/PC** to ensure that the F-CPU does not contain an "old" safety program:

1. Perform a general reset of the F-CPU using the **mode selector** or by means of the **programming device/PC.**
2. Download the configuration in *HW Config* to the F-CPU.
3. Download the safety program to the F-CPU in the "Safety Program" dialog box.
4. Run a program identification (in other words, check whether the collective signatures of all F-blocks with the F-attribute of the online and off-line block container match, see manual, Sections *5.8.4* and *5.10.2*).

Safety Note

If **more than one F-CPU** is accessible over a network (such as MPI) from **one programming device or PC**, you must take the following additional measures to ensure that the safety program is downloaded to the correct F-CPU:

Use passwords specific to each F-CPU, such as a uniform password for the F-CPUs having the respective MPI address as an extension: "Password_8".

Note the following:

- A point-to-point connection must be used the first time a password is assigned to an F-CPU (this also applies to the first time an MPI address is assigned to an F-CPU).
- Before downloading a safety program to an F-CPU that has not yet been assigned access protection with an F-CPU password, you must first revoke existing access permission for any other F-CPU.

1.2.17.2 Transferring the Safety Program to the F-CPU Using a Memory Card

Use of MMC or Flash Card

The following safety note applies to use of the following:

- Flash card (for example with CPU 416F-2)
- MMC (for example with CPU 317F-2 DP, CPU 315F-2 DP or IM 151-7 F-CPU)

Safety Note

If the function of the safety program is not tested in the target F-CPU, you must comply with the following procedure when **transferring the safety program to the F-CPU using a memory card** (MMC of Flash Card) to ensure that the F-CPU does not contain an "old" safety program:

1. Turn off the power to the F-CPU and remove the battery of F-CPUs with battery backup (for example with the CPU 416F-2). (To make sure that the F-CPU is no longer powered, wait for the buffer time of the power supply you are using or, if this is unknown, remove the F-CPU.)
2. Remove the memory card (MMC or Flash Card) containing the old safety program from the F-CPU.
3. Insert the memory card (MMC or Flash Card) containing the new safety program in the F-CPU.
4. Turn on the power to the F-CPU again and insert the battery of F-CPUs with battery backup (for example with the CPU 416F-2).

You must make sure that the inserted memory card (MMC or Flash-Card) contains the correct safety program. You can do so through a program identification or other measures, such as a unique identifier on the memory card (MMC or Flash Card).

When **downloading a safety program to a memory card** (MMC or Flash Card), you must use the following procedure:

1. Download the safety program to the memory card (MMC or Flash Card).
2. Run a program identification (in other words, check whether the collective signatures of all F-blocks with the F-attribute in the off-line block container and on the memory card (MMC or flash card) match, see manual, Section *5.10.2*).
3. Label the memory card (MMC or Flash Card) accordingly.

The procedure outlined must be ensured through organizational measures.

1.2.18 Deactivating Safety Mode

Safety operating mode/evaluating deactivated safety mode

If you wish to evaluate the safety operating mode or deactivated safety mode in the safety program, you can evaluate the "MODE" tag in the F-shared DB (1 = deactivated safety mode).
These tags are accessed fully qualified ("F_GLOBDB".MODE). The number and symbolic name of the F-shared DB and the absolute address of the tags are indicated in the printout of the safety program.

You can use this, for example, to passivate F I/Os when the safety program is in the deactivated safety mode. To do this, assign the "MODE" tag in the F-shared DB to all "PASS_ON" tags in the F I/O DBs of the F I/Os that you wish to passivate.

Safety Note

When the safety program is in the deactivated safety mode, the evaluation of the "MODE" tag in the F-shared DB is performed in deactivated safety mode.

Even when the F I/O in the deactivated safety mode is passivated by the evaluation of the "MODE" tag, due to the deactivated safety mode the safety of the plant must be ensured by other organizational measures, such as monitored operation and manual safety shutdown.

Please refer to the safety note in the manual, Chapter *5.9.1*.

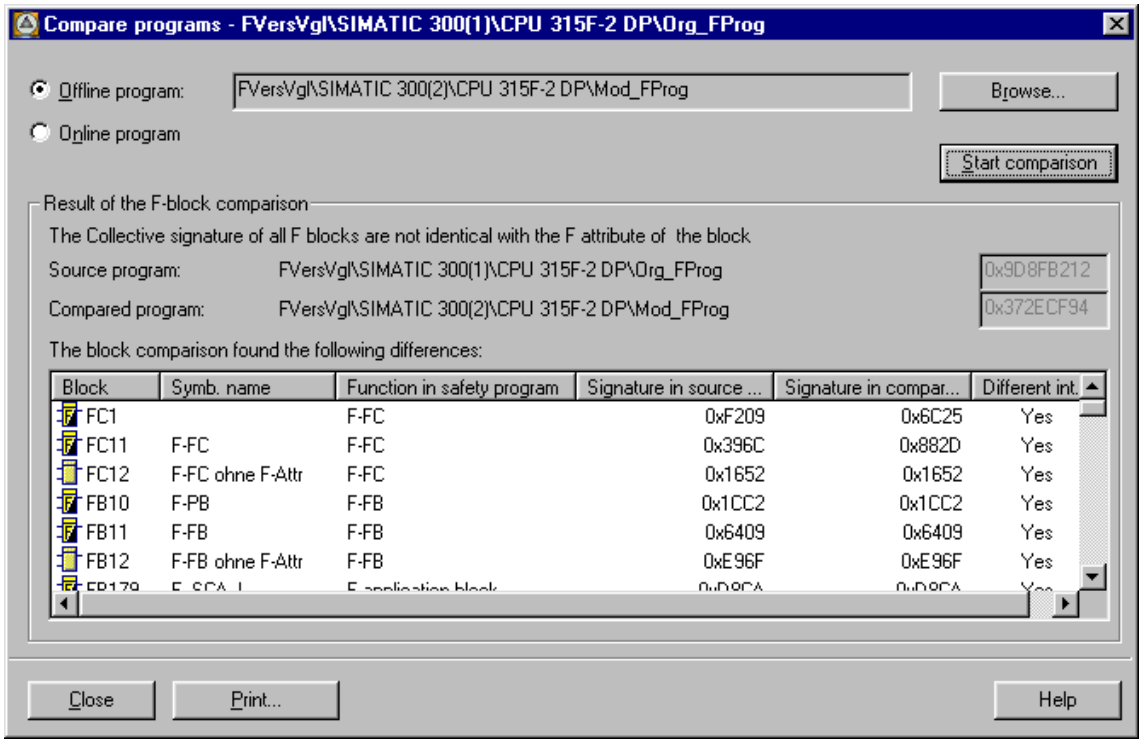
1.2.19 Comparing Safety Programs

"Compare Program" Dialog Box

The following information supplements Section *5.10.2* of the manual.

The "Compare Program" dialog box has been expanded to include the following two columns:

- "Function in the Safety Program"
- "Interface Different"



The "**Function in the Safety Program"** column displays the function of the F-blocks in the safety program.

The **"Interface Different"** column displays whether or not changes have resulted in the declaration table of F-blocks.

1.2.20 Printing Out Project Data of the Safety Program

Printing Out Additional Project Data with *S7 Distributed Safety* V 5.2 + SP 1

In addition to the project data indicated in Section *5.11* of the manual, the following project data are printed out for the safety program:

- The version of the F-compiler that was used to compile the safety program is printed out as an internal version ID.
- For F_GLOBDB: absolute and symbolic address of the time of compilation and of RLO 0 and RLO 1.
- The time of compilation of the safety program is printed out.
- The version of the F-compiler used to create the printout is indicated in the footer.
- A note is printed out if the amount of local data reserved for the safety program has been exceeded.

2 Corrections to the *S7 Distributed Safety, Configuring and Programming* Manual, A5E00109537-01, Edition 03/2002

Introduction

This section presents corrections to the above-indicated edition of the manual that could not be made prior to publication. The corrections are associated with the corresponding sections of the manual.

The corrections apply to versions V 5.1 and V 5.2 +SP 1 of the optional package *S7 Distributed Safety*.

Use of Software Packages with Standard User Program

For software packages that can be used in parallel for the standard program and safety program (for example, SW Redundancy), general conditions may apply that must be observed:

Note

If the safety program assigns block numbers (for FBs, DBs, and FCs) that are required by the software package, it may be necessary to change the safety program to release the block numbers when the software package is subsequently used. Changes made to the safety program must undergo a new acceptance test (see manual, Section *6.3*).

Section 3.3, Changing Safety-Relevant Parameters

Note

If you change a safety-relevant parameter for an F I/O, a fail-safe DP standard slave, or an F-CPU, you must recompile the safety program.

Section 3.4, Configuring the F I/Os

Safety Note

The switch setting on the address switch of the F I/O, in other words, its PROFIsafe target address must be unique within the network* and station** (throughout the system). You can assign a maximum of 1022 PROFIsafe target addresses in a system, in other words, a maximum of 1022 F I/Os can be addressed over PROFIsafe.
Exception: In different I-slaves, F I/Os can have the same PROFIsafe target address since they are only addresses within the station, in other words by the F-CPU in the I-slave.

The following restriction applies only to F-submodules ET 200S or fail-safe DP standard slaves whose default PROFIsafe addresses **cannot** be modified in *HW Config*:

If you use F-submodules ET 200S or fail-safe DP standard slaves in a PROFIBUS network, whose PROFIsafe addresses cannot be modified in *HW Config*, you can only operate **one DP master with F-CPU** in this network, otherwise the system-wide uniqueness of the PROFIsafe addresses cannot be guaranteed.

* A network consists of one or more subnets. "Throughout the network" means beyond the boundaries of PROFIBUS subnets.

** "Station-wide" means a station in *HW Config* (for example an S7-300 station or an I-slave)

Section 3.7, Configuring safety-related CPU-CPU Communication via DP/DP-Coupler (Master to Master Communication)

Note

If you intend to pass **more than 2 bi-directional** or **more than 4 unidirectional** communication connections through a DP/DP coupler, then you must use:

- 4 universal modules for each F-CPU per bi-directional communication connection
- 2 universal modules for each F-CPU per unidirectional communication connection

In the object properties of the DP/DP coupler, select "Output" and "Input" individually as I/O type instead of "Input/Output". Enter the values for the output data area in the universal module with I/O type "Output" and the values for the input area in the universal module with I/O type "Input".

If you follow the instructions as in the manual, Section *3.2*, you can currently operate a maximum of 2 bi-directional or 4 unidirectional communication connections over a DP-DP coupler.

Section 5.1.3, Supported Data and Parameter Types

Note

TIME input/output parameters of a calling F-FB/F-FC may

not be assigned as the current parameters for formal parameters of a called F-FB/F-FC.

In the safety program, read and write access to block parameters of the data type TIME in instance DBs of F-FBs is not permitted.

Section 5.1.3, Using the Local Data Address Area

Note

Note when using the local data address area that the first access of local datum in an F-PB/F-FB/F-FC must always be write access in order to initialize the local datum.

Take care that the initialization of the local datum is **not** skipped over by JMP, JMPN or RET operations (branched).

The initialization of a "local data bit" should be performed with the operational instruction ("=") (F-FUP) or relay coil, output ("--()") (F-KOP). Assign the local data bit signal state "0" or "1". Signal state "0" or "1" can also be obtained from the "VKE0" or "VKE1" tags in the F-shared DB through fully qualified DB access ("F_GLOBDB".VKE0 or "F_GLOBDB".VKE1).

Local data bits cannot be initialized with operations flip-flop (SR, RS), set output (S) or reset output (R).

The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"

Section 5.1.3, Illegal Address Areas

Note

Data from the standard user program (memory bits or PII of standard I/O) must not be used for edge memory bits of the query edge (N, P) or query signal edge (NEG, POS) operations, or for the address of the flip-flop (SR, RS) operations, since these data are read and written by this operation.

If the "local data" address area is used for the edge memory bits of the operations query edge (N, P) or query signal edge (NEG, POS) or for the address of the operations flip-flop (SR, RS), set output (S) or reset output (R), the local data bit used must be initialized beforehand.

Section 5.1.3, Access to Formal Parameters of an F-FB/F-FC

Note

Note that you can only read the input parameters in an F-FB/F-FC and only write to its output parameters.

Use an input/output parameter if you wish to read and write.

If you wish to use a formal parameter of an F-FB/F-FC for the edge memory bits of the operations query edge (N, P) or query signal edge (NEG, POS) or for the address of the operations flip-flop (SR, RS), set output (S) or reset output (R), this must be declared as an input/output parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"

Section 5.1.3, Access to data of an instance DB for which the corresponding F-FB call is not programmed

You must not access data of an instance DB in the safety program if the corresponding F-FB call is not programmed.

Section 5.1.3, NEG_I Operation

Note

If the result of an ADD_I, SUB_I, MUL_I, or **NEG_I** operation or the quotient of a DIV_I operation is outside the permitted range for integers (16 bits), the F-CPU changes to STOP if the result/quotient is used in an output to an F I/O or to a partner F-CPU over safety-related CPU-CPU communication. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"

You should make sure that the values remain within the permitted range for integers (16 bits) when writing your program!

Section 5.1.3, DIV_I Operation

Note

If the divisor (input IN2) of a DIV_I operation = 0, the quotient of the division (result of division at output OUT) = 0. The result behaves like the same operation in a standard user program. The F-CPU does **not** change to STOP. This is the response regardless of whether there is OV-bit scan programmed in the next network.

Section 5.3.2, F I/O DB

The description of the IPAR_EN and IPAR_OK tags in the F I/O DB has changed as follows:

	Tag	Data Type	Function	Default
Tags that Can or Must be Described	IPAR_EN	BOOL	Tag for reassigning fail-safe DP standard slave parameters	0
Tags that Can Be Evaluated:	IPAR_OK	BOOL	Tag for reassigning fail-safe DP standard slave parameters	0

IPAR_EN

The tag IPAR_EN corresponds to the tag iPar_EN_C in the PROFIsafe bus profile, V 1.2.

Refer to the PROFIsafe bus profile, V 1.2 and the documentation on fail-safe DP standard slaves to know when you have to set/reset these tags during a fail-safe DP standard slave parameter reassignment.

Safety Note

Please note that the affected F I/O is **no** longer **passivated** as of *S7 Distributed Safety*, V 5.2 for IPAR_EN = 1.

If passivation should be continued for IPAR_EN = 1, you must set the tag PASS_ON = 1 in addition.

IPAR_OK

The tag IPAR_EN corresponds to the tag iPar_OK_S in the PROFIsafe bus profile, V 1.2.

Refer to the PROFIsafe bus profile, V 1.2 and the documentation on fail-safe DP standard slaves for how to evaluate these tags during a fail-safe DP standard slave parameter reassignment.

Section 5.3.3, Fully Qualified DB Access

In contrast to the information in the manual, Section *5.3.3* access to F I/O DBs of the F I/O of which no channel is used in the safety program, does not lead to an F-CPU STOP. The safety program can be compiled.

Section 5.3.4 to 5.3.7, Signal Chart Figures

The signal charts presented in the "Signal Chart ..." figures in Sections *5.3.4* to *5.3.7* of the manual represent typical signal charts for the indicated behavior.

Actual signal charts and, in particular, the relative position of the status change of individual signals can deviate from the given signal charts within the scope of known distortion for cyclic program execution, depending on the following:

- Type of F I/O used
  (F I/O with inputs, F I/O with outputs, F I/O with inputs and outputs, S7-300 F-SMs, ET 200S F-modules, or fail-safe DP standard slaves, version of PROFIsafe bus profile for the F I/O)
- Cycle time of OB of safety program
- Target rotation time of PROFIBUS-DP

Note

The signal charts refer to the status of signals in the user's safety program. If the signals are evaluated in the standard user program before or after the safety program is called in the same OB, the status change of the signals can be displaced by one cycle.

Contrary to what is shown in the status charts, status changes between process and fail-safe values that are transmitted to the fail-safe outputs ("To Outputs" signal chart) can occur before the status change of the associated QBAD signal, if necessary. The timing of the status change is dependent on whether F I/O with outputs or F I/O with inputs and outputs were used.

Section 5.3.8 omitted

Section 5.4, Programming of Safety-Related Communication by Means of DP/DP Coupler

---

Note

A separate instance DP must be used for each call of an F SENDDP or F_RCVDP block.

Section 5.5, Data Transfer from Safety to Standard User Program

Note

The process image input table of the F I/O is updated not only at the start of the F run-time group before processing the F-program block, but also by the standard operating system.

You can find out the time at which the PII is updated by the standard operating system in the *online help for STEP 7*, "Process image inputs/outputs". With the S7-400, remember the update times when using process image partitions. When accessing the process image input table of F-I/Os in the standard user program, you can therefore obtain other values than in the safety program. The differing values can result as follows:

- due to the different update points
- due to use of substitute values in the safety program

To obtain the same values in the standard user program as in the safety program, you can therefore only access the process image input table in the standard program after processing the F run-time group (F-CALL). In this case, you can also evaluate the QBAD tag in the relevant F I/O DB in the standard user program to find out whether the process image of the inputs contains substitute values (0) or process values. When using process image partitions (S7-400 only) make sure that between the processing of the F run-time group (F-CALL) and evaluation of the process image input table in the standard user program, there is no update of the process image by the standard operating system or by SFC 26 UPDAT_PI.

Section 5.6, Work Memory Requirements of the Safety Program

You can estimate the work memory requirements of the safety program as follows:

Work memory requirements for the safety program:

26 Kbytes for F-system blocks F_CTRL_1, F_CTRL_2 and F_IO_BOI

\+ 4.5 x work memory requirements of all F-FB/F-FC/F-PB

\+ 4.5 x work memory requirements of all F application blocks used (except F_SENDDP and F_RCVDP)

\+ work memory requirements of F application blocks used F_SENDDP and F_RCVDP (each 4.4 Kbytes)

Work memory requirements for data:

5 x work memory requirements of all F-DBs and I-DBs for F-PB/F-FB

\+ 2.3 x work memory requirements of all I-DBs for F application blocks (except F_SENDDP and F_RCVDP)

\+ work memory requirements of all I-DBs of the F application blocks F_SENDDP and F_RCVDP

\+ 0.7 Kbytes per F I/O (for F I/O DBs)

Block size of all automatically generated F blocks

To make sure that the automatically compiled F-blocks do not exceed the maximum possible size in the particular F-CPU, remember the following:

- The maximum size of an F-FB/F-FC/F-PB should be a quarter of the max. size of the FBs or FCs (see *Technical Specifications in the manual of the F-CPU you are using*).
- For F-FBs (including the F-PB), the following must apply:

  2 x number of all parameters or static data of the data type BOOL

  + 4 x number of all parameters or static data of the data type INT

  + 6 x number of all parameters or static data of the data type TIME

  < maximum size of the data blocks in bytes
  (see *Technical Specifications in the manual of the F-CPU you are using*)
- The following must apply to F-DBs:

  2 x number of all tags of the F-DB of the data type BOOL

  + 4 x number of all tags of the F-DB of the data type INT

  + 6 x number of all tags of the F-DB of the data type TIME

  < maximum size of the data blocks in bytes
  (see *Technical Specifications in the manual of the F-CPU you are using*)

If you receive the message "The block x could not be copied" when you download your safety program to the F-CPU, check whether these conditions are met and reduce:

- the size of the F-FB/F-FC/F-PB or
- the number of parameters and static data of the F-FBs/F-PBs or
- the number of tags of the F-DBs.

Section 5.6.3, "Inserting Application Templates"

Prior to inserting the application template F_ESTOP (FBD) or F_ESTOP (LAD), you must copy the F-application block F_TOF from the block container F-Application Blocks\Blocks of Distributed Safety F-Library (V1) to the block container of your S7 program, if it is not already present.

Section 5.7.3, F Application Blocks

Safety Note

You must use a separate instance DB for each call of one of the following F application blocks:

F_TP, F_TON, F_TOF, F_ACK_OP, F_2HAND, F_MUTING or F_1oo2DI

Each call for the F application blocks listed must only be processed once in a cycle of the F run-time group.

Section 5.7.3.8, F_2HAND: Two-Hand Monitoring

The following paragraph replaces the note on the functionality of F_2HAND in the manual:

Note: With an F application block, only one signal per button can be evaluated. With suitable configuration (type of sensor wiring: antivalent sensor) the discrepancy monitoring of the NC and NO contact of the IN1 and IN2 button is performed directly by the F I/O with inputs. If discrepancy is detected, the substitute value 0 is entered in the process image input table (PII) and QBAD is set to **1** in the relevant F I/O DB.

Section 5.7.3.9, F_MUTING

Safety Note

The muting lamp must be monitored by means of input QBAD_MUT. To do this, you must wire the muting lamp to an output with the wire break monitoring of an F I/O and connect the QBAD signal of the associated F I/O DB to input QBAD_MUT.

The only suitable F I/Os are those that monitor wire breaks every 200 ms after activation of muting (for example SM 326; 10 x DC 24V/2A). The F submodules PM-E F DC24V PROFIsafe and 4 F-DO DC24V/2A PROFIsafe are currently not suitable.

Section 5.9, Test Options

Modifying data of the safety program using "Monitor/Modify Tags" and write access using *HW Config* or the *FBD/LAD editor* are possible only with restrictions and with the safety mode deactivated.

Section 5.9.2, More rules for testing the safety programs

In addition to the information in the manual, Section *5.9.2* setting breakpoints in the standard user program can lead to the following error in the safety program:

- internal CPU error