Contents 13

13. Organization Blocks

At the end of the chapter the participant will ...

... be familiar with the different types of organization blocks

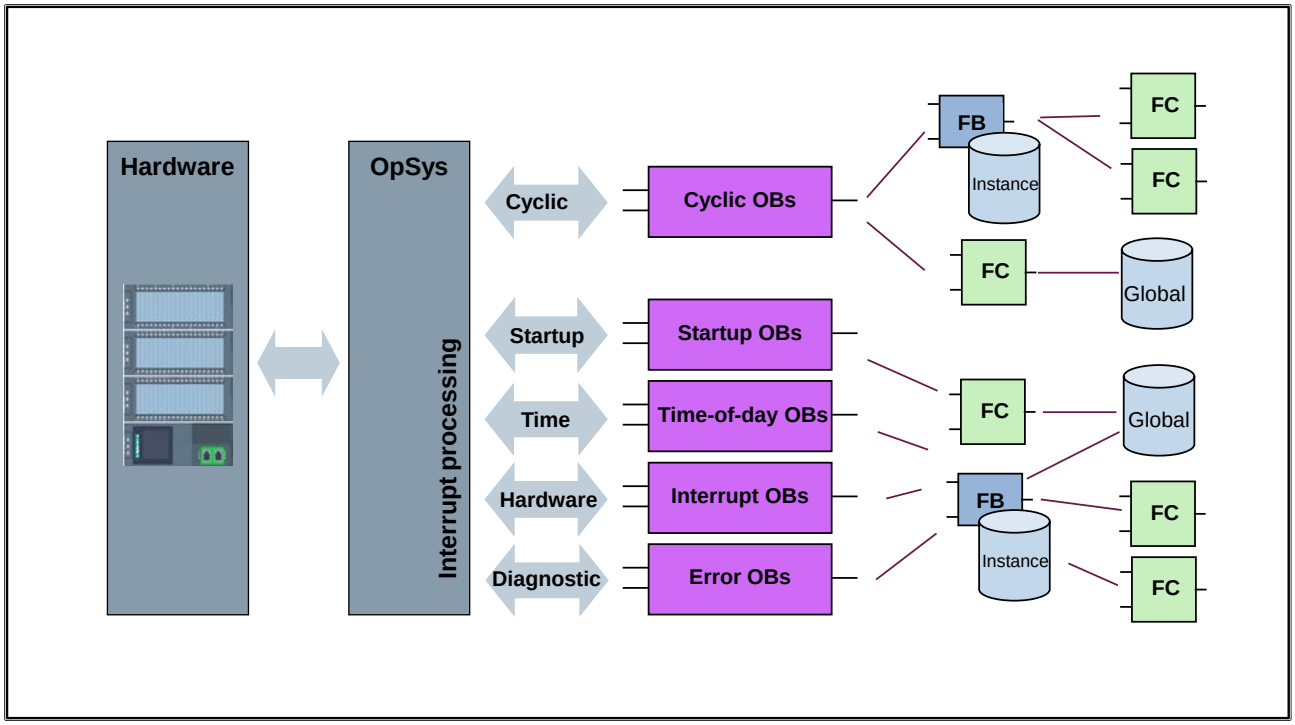
... understand the principle of interrupt processing

... be familiar with the meaning of process image partitions

... be able to interpret the start information of OBs

… be able to use startup and time-delay interrupt OBs

13.1. Types of Organization Blocks



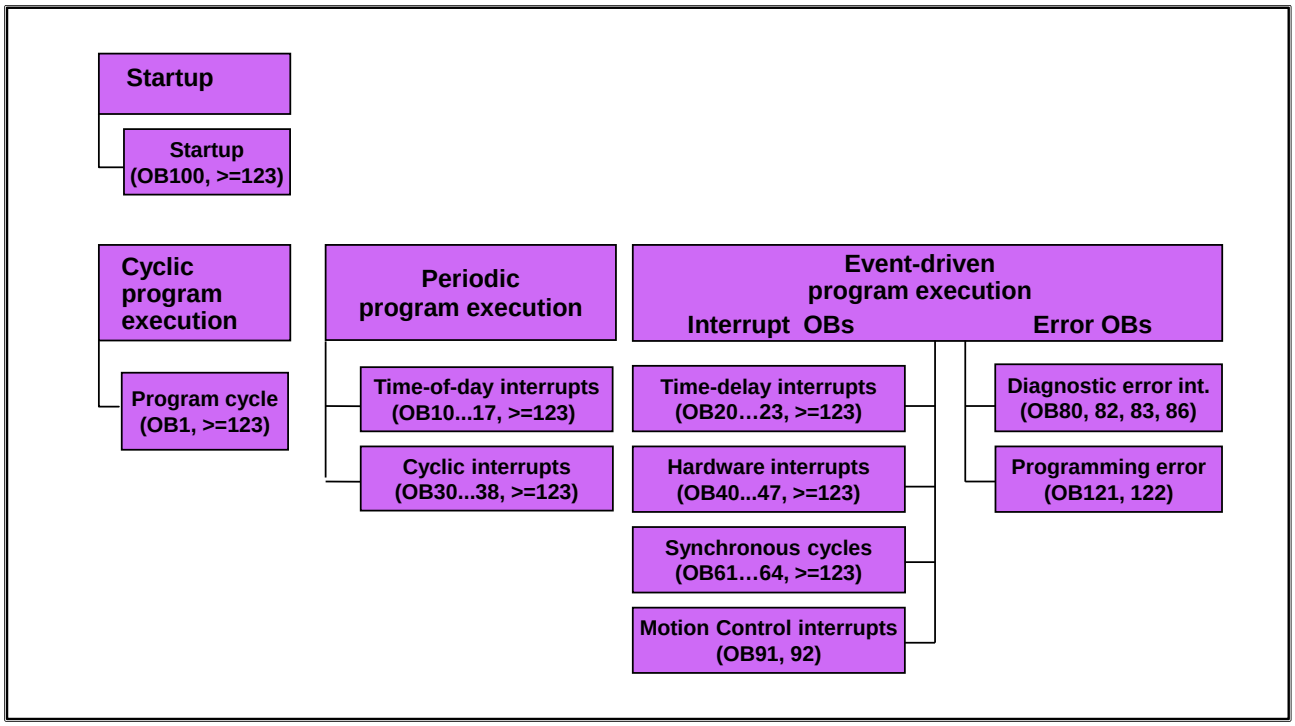
Organization Blocks (OBs)

Organization blocks (OBs) form the interface between the operating system and the user program. The organization blocks are called event-driven by the operating system.

The events can be cyclic, the STOP-RUN transition, time-dependent, hardware-dependent or an error.

If one of the events occurs, then the relevant OB is called if it is loaded and if its priority is greater than that of the OB currently being processed.

13.1.1. Organization Blocks of the S7-1500



Startup Program

After voltage recovery, or a change of operating mode (through the CPU‘s mode selector or through PG operation), a startup program is carried out in the Startup OBs before the cyclic program execution. In these Startup OBs you can, for example, carry out a pre-assignment of communication connections or initializations.

Cyclic Program Execution

The program stored in the Program Cycle OBs is executed in a continuous loop. With this cyclic program execution, the reaction time results from the execution time for the CPU’s operating system and the sum of the command runtimes of all executed instructions. The reaction time, that is, how fast an output can be switched in relation to an input signal, amounts to a minimum of one time and a maximum of two times the cycle time.

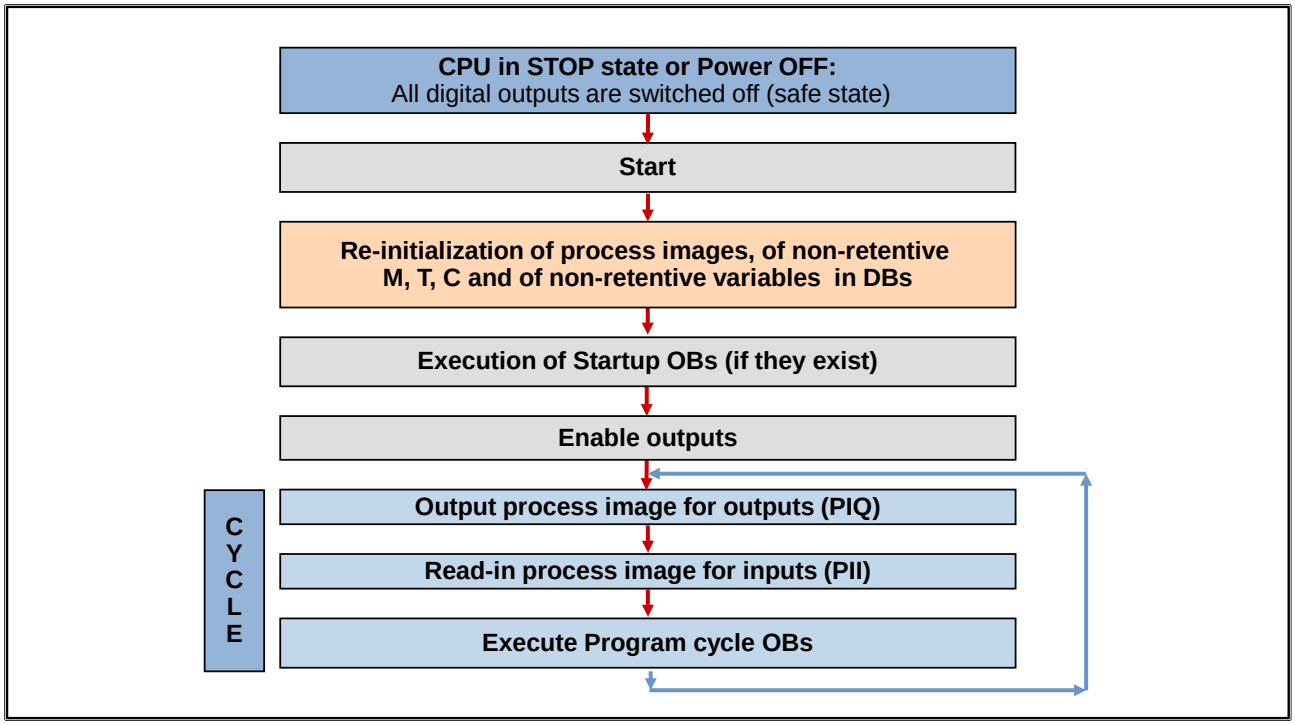
Periodic Program Execution

This makes it possible to interrupt the cyclic program execution at fixed intervals. With the Cyclic Interrupt OBs, an organization block (for example OB35) is executed after an adjustable time base (for example, every 100ms) has expired. In these blocks, for example, closed-loop control blocks with their sampling time are called. With the Time-of-day Interrupt OBs, an OB which could carry out a data backup, for example, is executed at a specific time, for example, every day at 17:00 hours (5 p.m.).

Event-driven Program Execution

Hardware interrupts are used to quickly react to process events. After an event occurs, the cycle is immediately interrupted and an interrupt program is executed. With Time-delay Interrupt OBs, a freely definable event can be reacted to with a time-delay; with the Error OBs, the user can influence the behavior of the controller in case there is an error.

13.2. S7-1500 Start and Cyclic Sequence



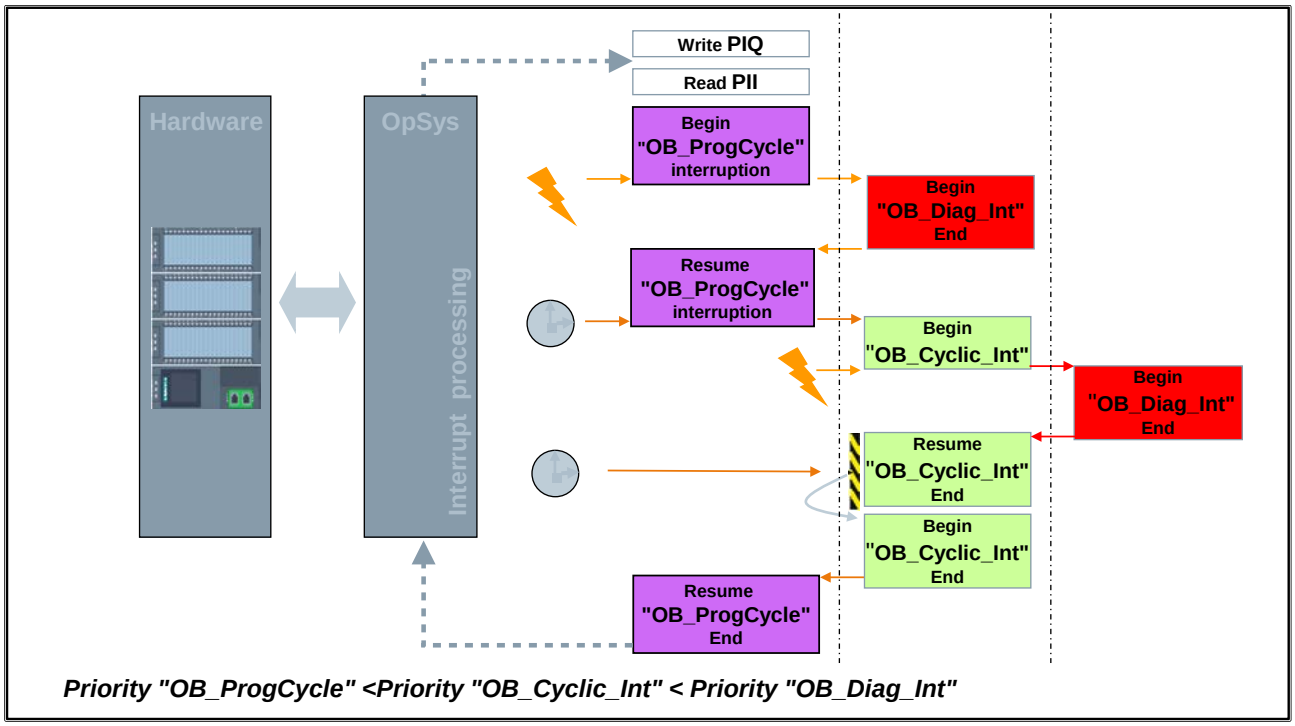
General

Before the CPU begins with the execution of the cyclic user program, a startup program is executed.
In the startup program, initialization variables can be set by programming Startup OBs accordingly.

Warm Restart

The S7-1500 carries out a so-called warm restart in which the process images (PII, PIQ) and all non-retentive memory bits, timers and counters are deleted. Non-retentive DBs are reset to the start values of the load memory and retentive memory bits, timers and counters as well as retentive DB-contents are retained.

13.2.1. Interrupting the Cyclic Program



OB Calls

Organization blocks (OBs) form the interface between the CPU's operating system and the user program.
Organization blocks are called exclusively by the operating system. There are various start events (time-of-day interrupts, hardware interrupts - see picture) that each lead to the start of their associated organization block.

Interrupting the Cyclic Program

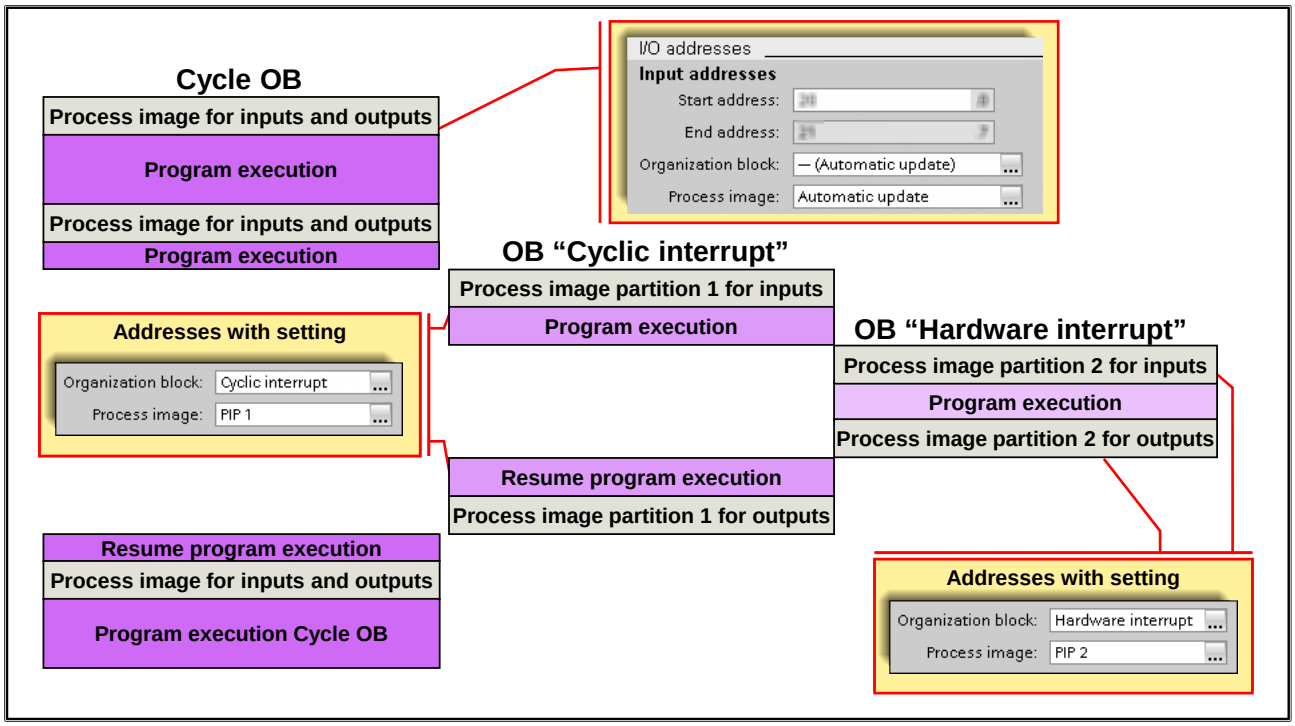
When the operating system calls another OB, it interrupts the cyclic program execution because a "Program Cycle"-OB has the lowest priority. Any other OB can therefore interrupt the main program and execute its own program. Afterwards, the "Program Cycle"-OB resumes execution at the point of interruption.

Priorities

The S7-1500 CPUs support the priorities 1 (lowest priority) to 26 (highest priority).
The OBs are executed on a purely priority-driven basis. This means that when several OB requests exist at the same time, the OB with the highest priority is executed first. When an event occurs that has a higher priority than the currently active OB, this OB is interrupted. Events of the same priority are executed in the order that they occur.

If this is also the same, for example for Startup OBs, then the OBs are executed according to their number.

13.2.2. Process Image Partitions



Process Image for Inputs and Outputs

For reasons of access speed and the consistency of the status of individual inputs throughout the entire cycle, a copy is stored in the process image for inputs (PII) which is accessed during program execution. The writing of outputs occurs in the process image for outputs (PIQ). The statuses of outputs stored here are written into the actual outputs at the beginning of every cycle.

The inputs and outputs of an S7-1500 are automatically made available in the process image for inputs and outputs as long as this is not changed in the settings of the individual blocks.

Under the Properties of the module > *General* > *Input…/Output…* > *I/O addresses,* "Automatic update" ("Automatische Aktualisierung") is set in *Organization block* and *Process image*.

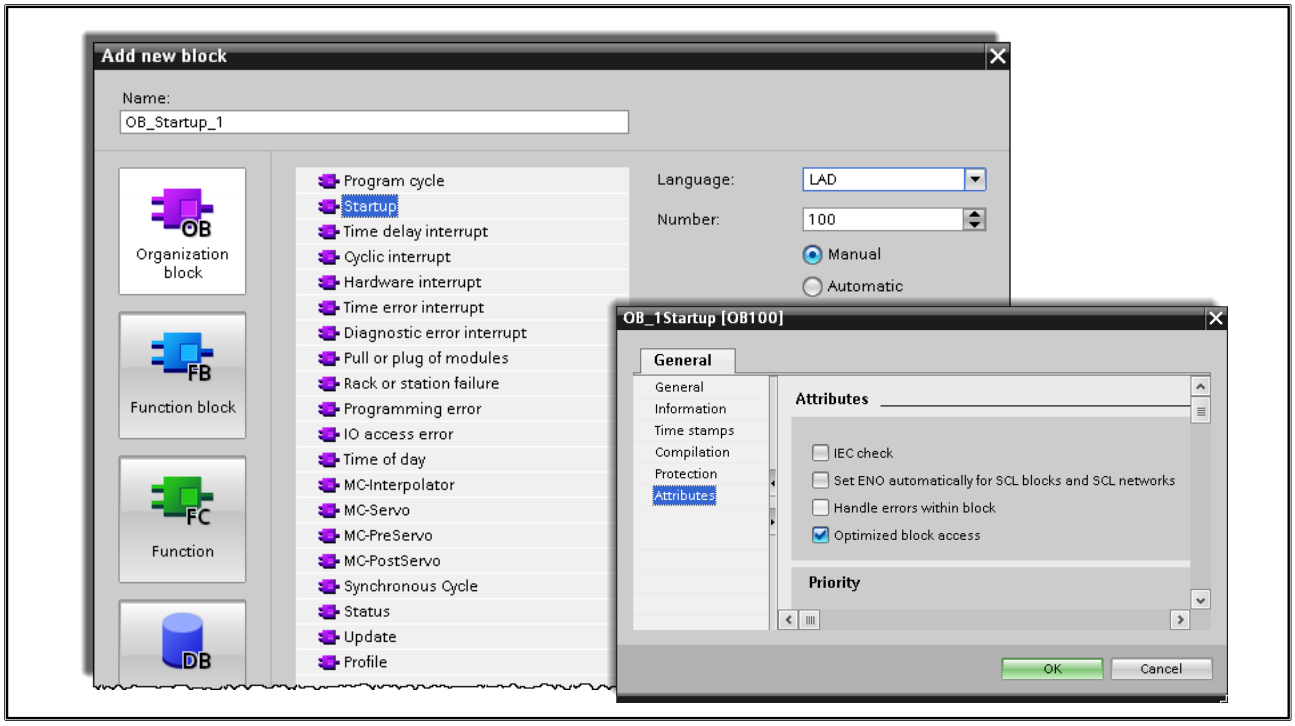
Process Image Partition (PIP)

Inputs and outputs which are not used in the cyclic program, that is, not in every cycle, do not have to be updated with the process image at the beginning of every cycle. For that reason, there are the process image partitions which in turn can be assigned to individual organization blocks. If an OB is executed which has a PIP assigned to it, then the inputs of the associated PIP are read-in at the beginning and at the end, the statuses of the outputs of the PIP are written in the relevant peripherals. If a process image partition is not assigned to any organization block, then the update of this PIP must be done with the help of instructions.

Updating a Process Image Partition in the User Program

Each PIP can be updated in the user program with special instructions. For this, the instructions "UPDAT_PI" for updating inputs and "UPDAT_PO" for updating outputs is used.

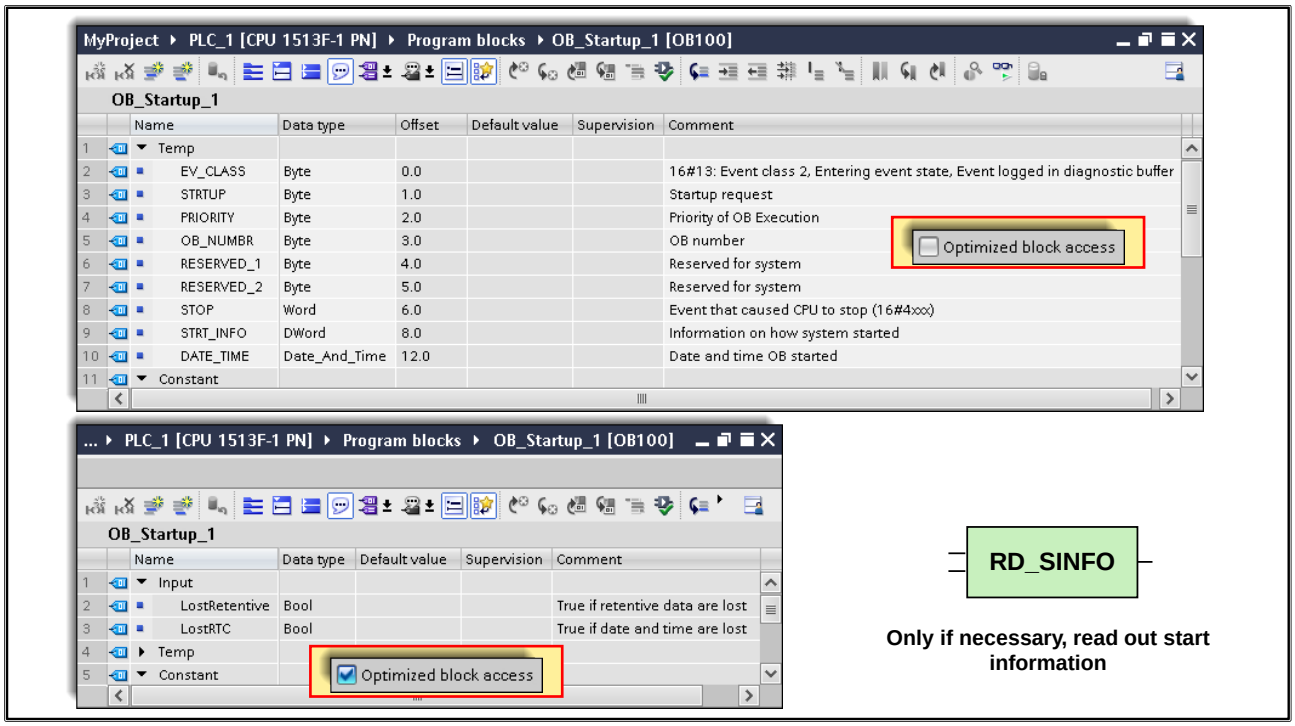
13.3. Creating a new OB



Create new OB

When creating an organization block, the type of event is first of all selected. In addition, the number and the name can be changed. By default, new OBs are created with the attribute "Optimized block access" with the result that only reduced start information is available in the OBs (see next page). In the Properties of the OB, the behavior can be individually set depending on the type.

13.3.1. OB Start Information using "OB_Startup" as an Example



Start Information for Not Optimized Block Access

When the operating system calls organization blocks, the user is provided with OB-specific start information on the local data stack.

This start information has a length of 20 bytes and is available after the OB starts execution.

The start information as well as their absolute L-stack addresses is only completely available for those OBs where the block attribute "Optimized block access" is **not** activated (as shown in the picture).
In order to avoid errors, the structure of the standard declaration section should not be changed by the user. Following the standard declaration section, the user can declare further, additional temporary variables.

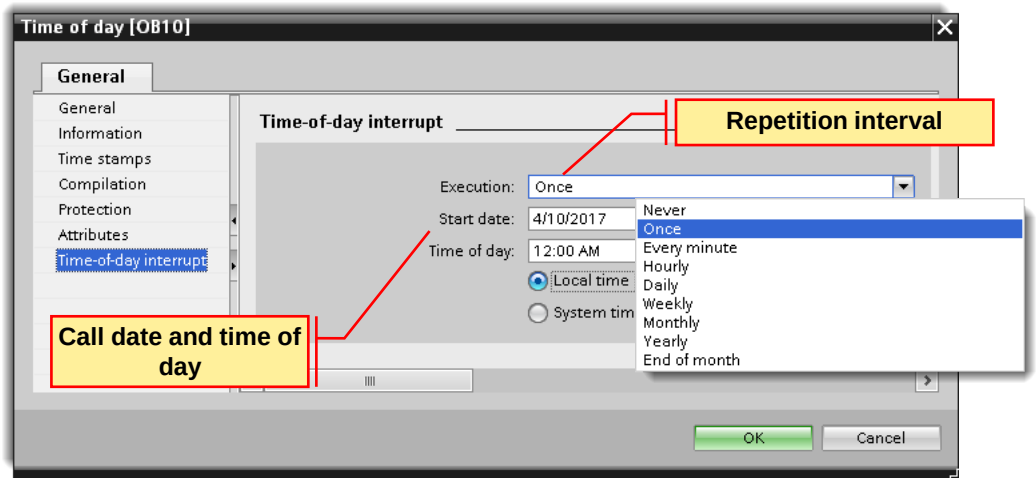
Start Information for Optimized Block Access

The start information of an organization block with optimized block access is limited to the essentials and is passed by means of input parameters. If necessary, the start information, with the exception of the date and time, can be read out by means of the function "RD-SINFO".

Variables

An explanation of the meaning of the variables can be found in the online help.

13.4. Time-of-Day Interrupt OB



Time-of-Day Interrupts

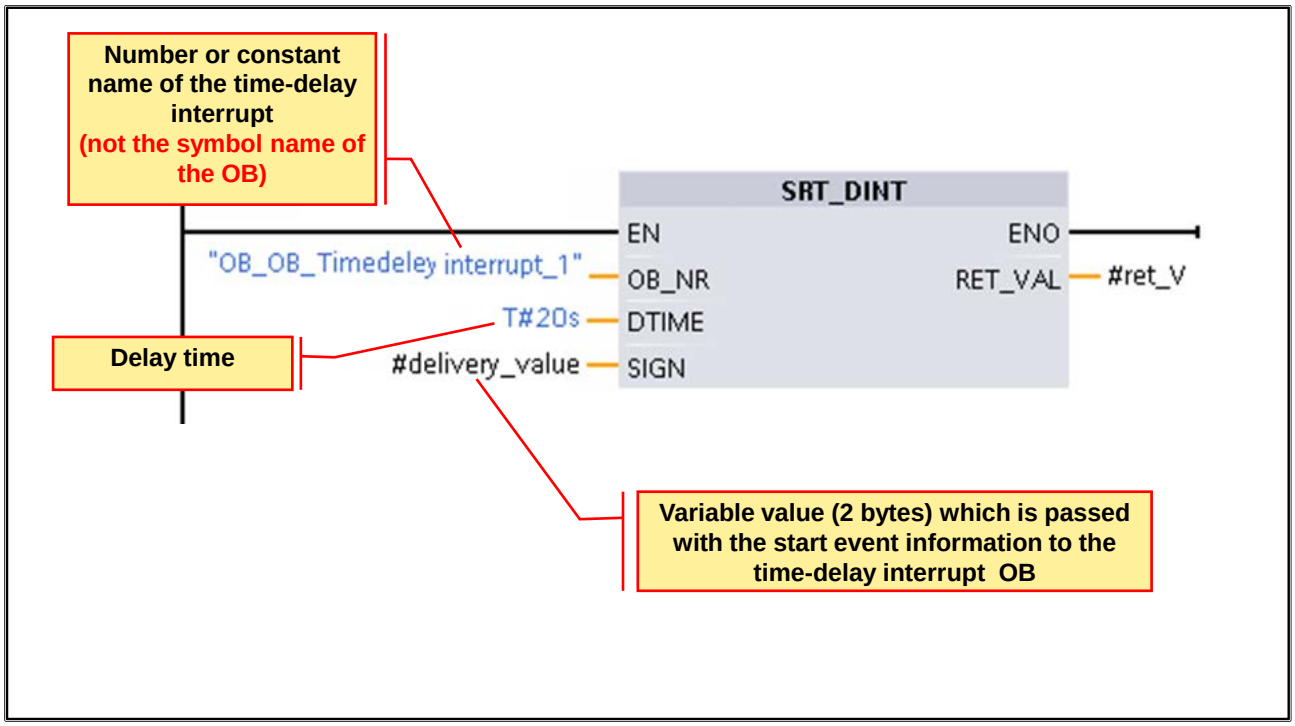
Time-of-day interrupts are used for executing a certain program called in OB 10 (as an example) either once only at a certain time or periodically (every minute, hourly, daily, weekly, monthly, yearly) starting at that time.

Note

In addition, the time-of-day interrupts can be controlled at runtime with the following instructions (Task Card Instructions "Extended instructions -> Interrupts > Time-of-day interrupt"):

- "SET_TINT" Set start date, time and period
- "SET_TINTL" Set start date, time and period
- "CAN_TINT" Cancel time-of-day interrupt
- "ACT_TINT" Activate time-of-day interrupt
- "QRY_TINT" Query time-of-day interrupt

13.4.1. Starting the Time-delay Interrupt OB



Time-Delay Interrupts

Time-delay interrupt OBs are used in order to be able to react to freely definable events after a time delay. With the function "SRT_DINT", you define which time-delay interrupt after expiration of which time is to be called by the operating system. With the help of the input parameter "SIGN", a value in the size of a word can be passed to the time-delay interrupt OB.

Caution:

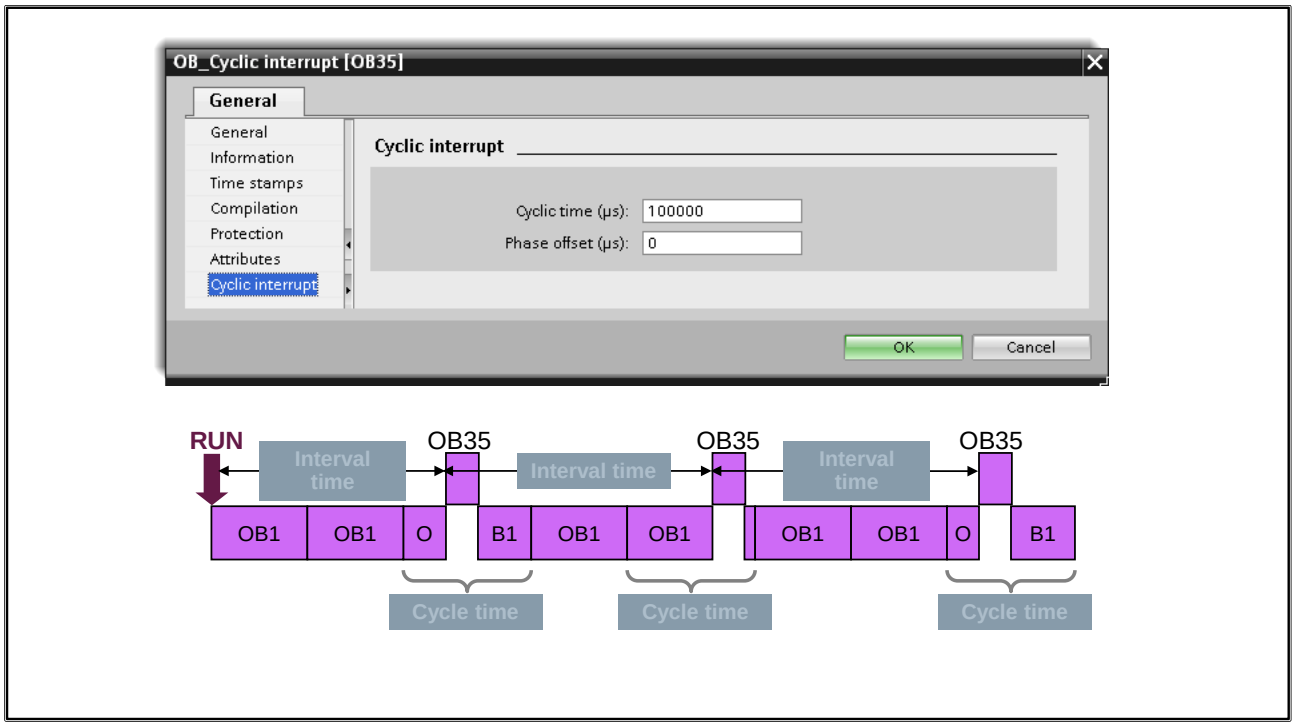
As the actual parameter of the formal parameter "OB_NR", the OB number or the constant name of the OB must be specified. The symbol name of the OB to be called is not acceptable.
You will wind the constant name in the Properties of the OB under "General > Constant name".

Note

In addition to the instruction "RSD_DINT", there are also other instructions in the Task Card Instructions under "Extended instructions -> Interrupts > Time delay instruction":

- "CAN_DINT" Cancel time delay interrupt
- "QRY_DINT" Query status of time delay interrupt

13.4.2. Executing Cyclic Interrupt OBs



Cyclic Interrupt

With a Cyclic interrupt OB, a block can be executed at fixed time intervals. The S7-1500 offers the OB 35, for example, as a Cyclic interrupt OB. The default setting for its call interval is 100000µs; the selectable range is from 500µs to 60000000µs (60sec).

Interval Time

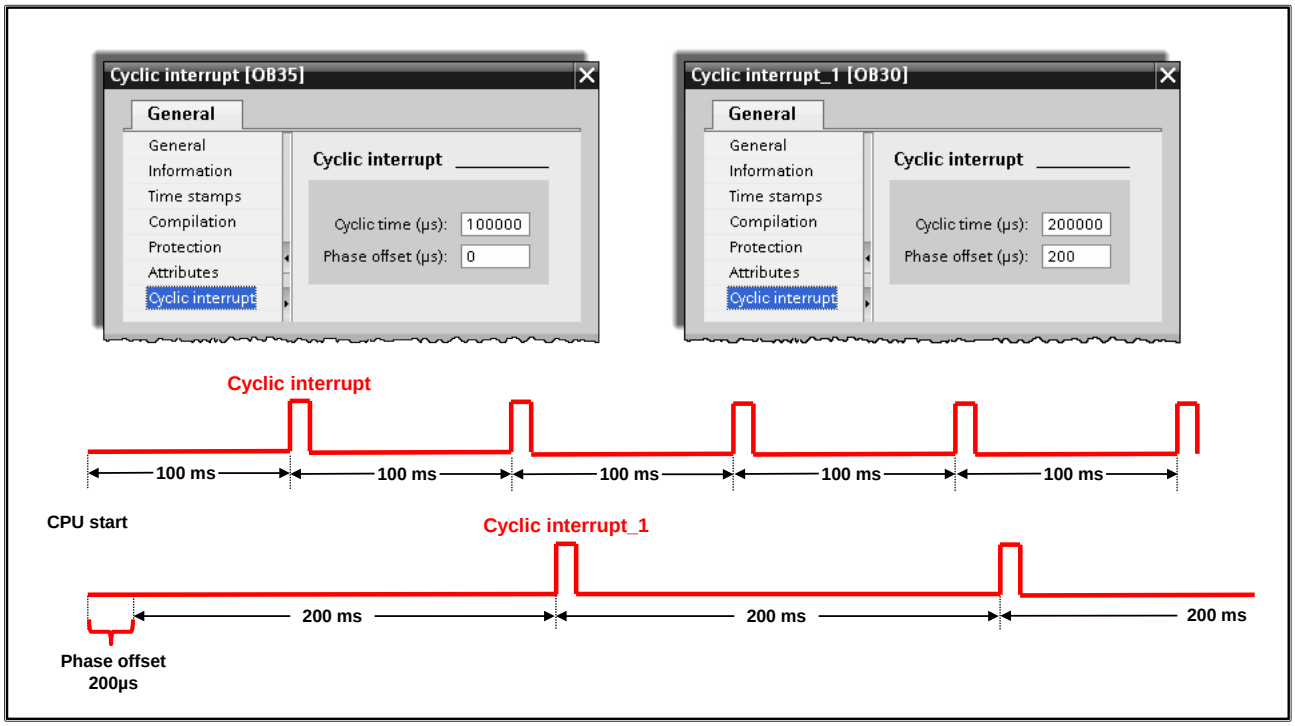
You must make sure that the interval you specify is longer than the time required for execution. The operating system calls the "Cyclic interrupt" OB at the specified time. If the "Cyclic interrupt" OB is still active at this time, the operating system calls the "Time error interrupt" (OB 80).

Note

Cyclic interrupts can also be controlled and queried at runtime with "Extended instructions" (Task Card Instructions "Extended instructions -> Interrupts > Cyclic interrupt"):

- "SET_CINT" Set cyclic interrupt parameters
- "QRY_CINT" Query cyclic interrupt parameters

13.4.2.1. Phase Offset for "Cyclic interrupt" OBs



Example for the Use of a Phase Offset

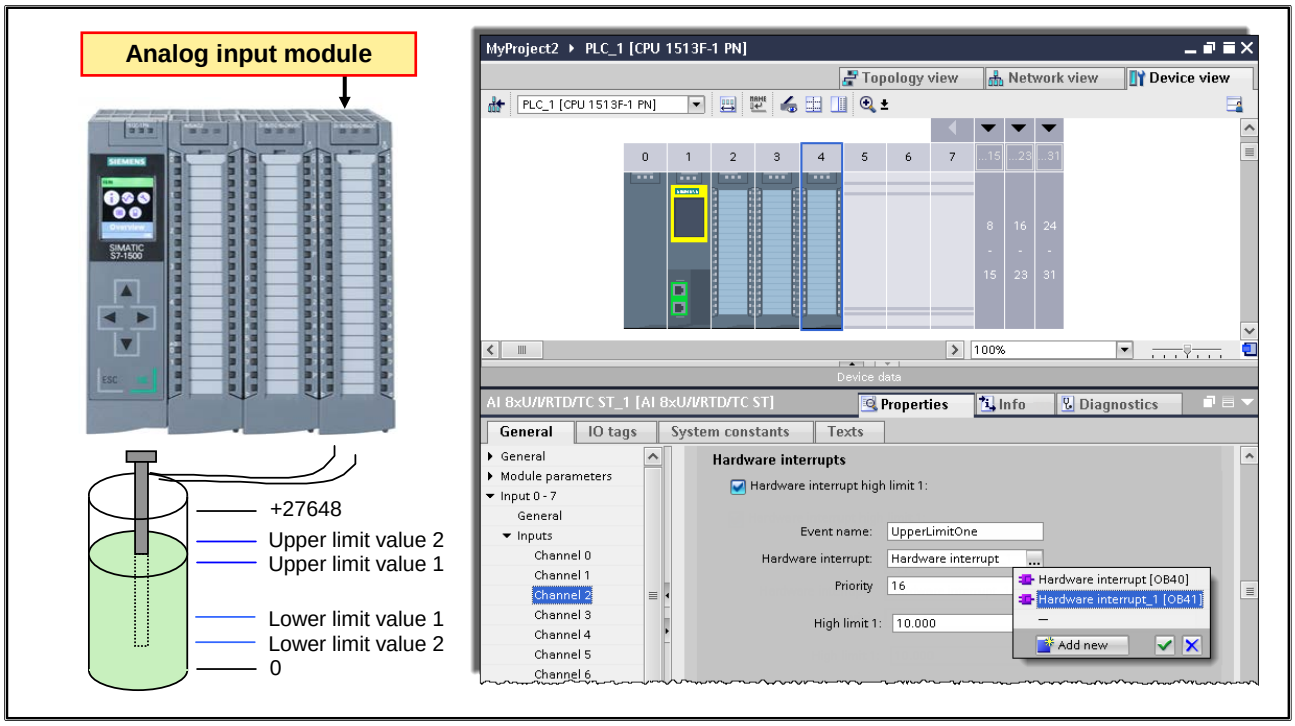
Two "Cyclic interrupt" OBs are required in the program:

"Cyclic interrupt"

"Cyclic interrupt_1"

For the OB "Cyclic interrupt" and for the OB "Cyclic interrupt_1", a time period of 100 ms was set. After the time period of 100 ms has expired, both "Cyclic interrupt" OBs are given their starting time. However, in order to execute the OBs with a time lag, a phase offset is configured for one of the two OBs.

13.4.3. Hardware Interrupt



Hardware Interrupt

The program execution of a "Hardware interrupt" OB is started as soon as a certain event occurs.

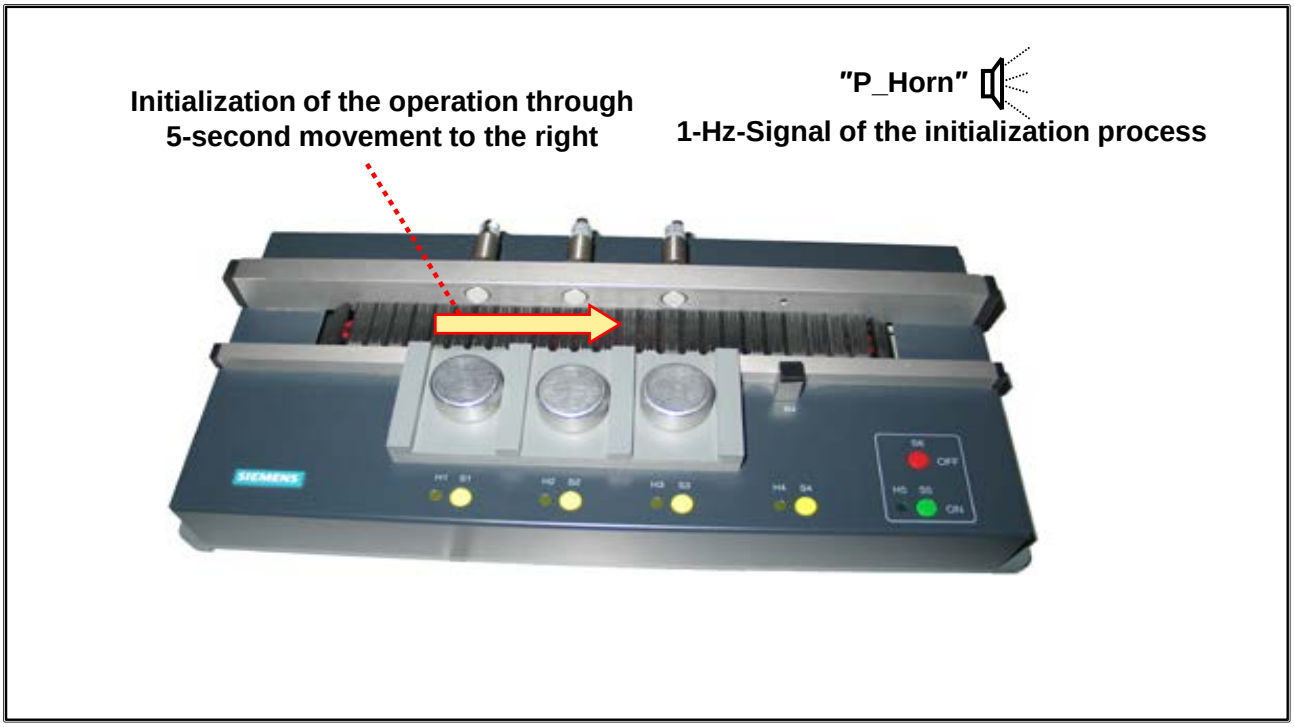
Hardware interrupts can be triggered by various module-specific signals:

For parameter-assignable signal modules (DI, DO, AI, AO) you specify which signal is to trigger the hardware interrupt in the Properties of the modules.

Example

In configuring an analog input module, suitable limit values were specified in the above example. If the measured value then exceeds this limit, an interrupt is triggered on the CPU which causes the program to be interrupted and the OB "Hardware interrupt" to be called for execution.

13.5. Task Description



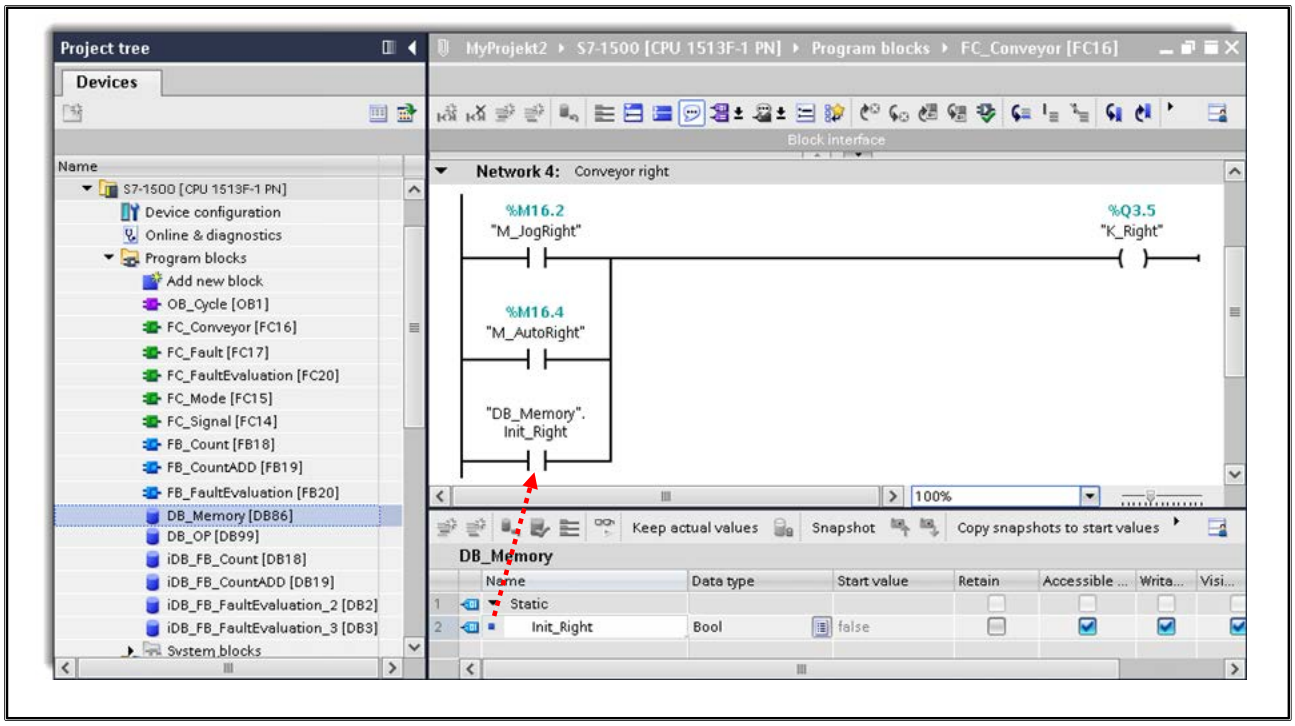
Task Description

In order to prevent parts being on the transport conveyor after switch on or warm restart, the conveyor is to move to the right for 5 seconds when there is a STOP-RUN transition.

For this, the conveyor is started with the help of a Startup OB during startup and is stopped again with the help of a Time-delay interrupt OB which is called after 5 seconds.

The initialization run is signaled with a 1 Hz signal at the horn.

13.5.1. Exercise 1: Preparing the Startup Initialization

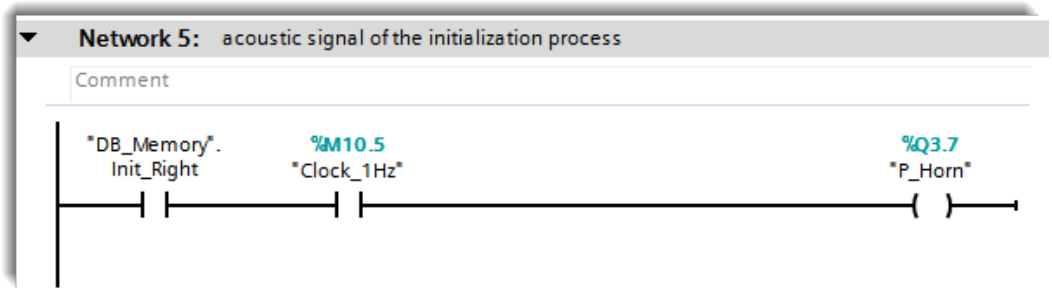


Task

So that the conveyor moves to the right after starting, a variable is required which has the Status True during the initialization time.

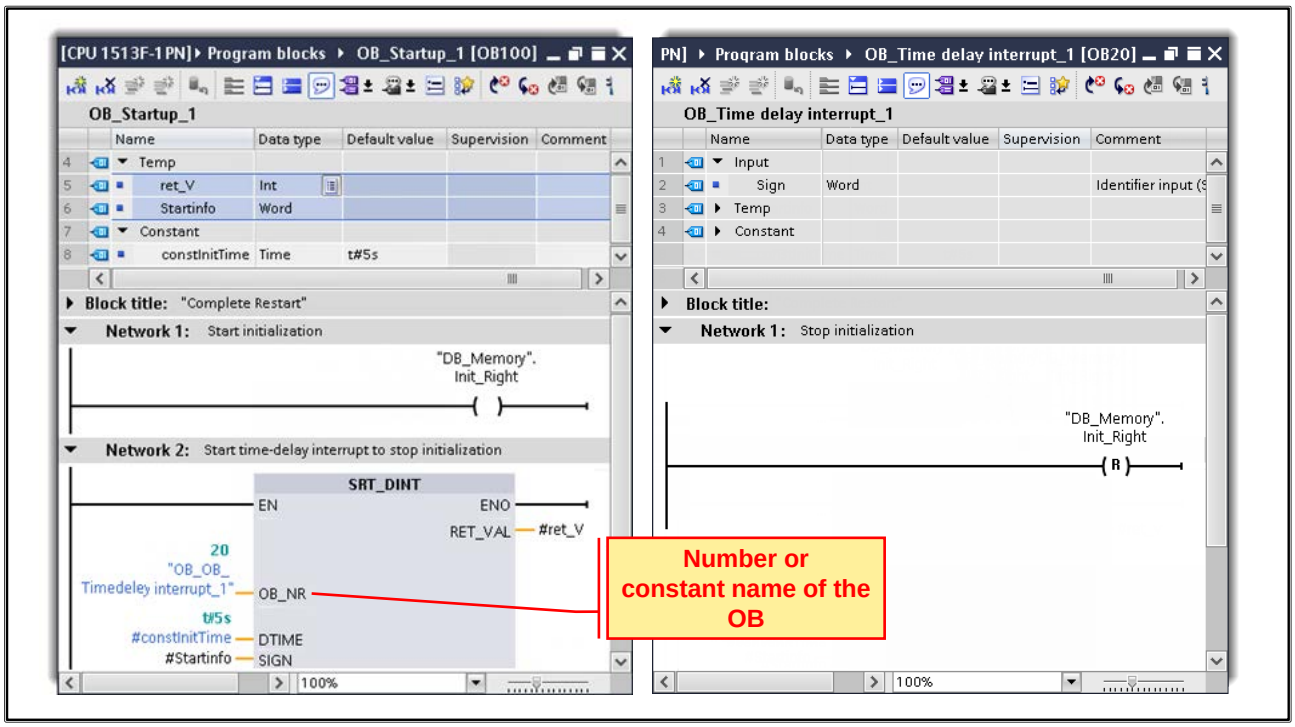
What to Do

1. Insert the new data block "DB_Memory" and declare the variable "Init_Right".
2. Link the variable "DB_Memory".Init_Right in "FC_Conveyor" as an additional OR-condition for transport conveyor movement to the right.
3. Program the acoustic signal of the initialization process in "FC_Signal". (see picture).



4. Save your project.

13.5.2. Exercise 2: Initializing Transport using Startup and Programming the Time-delay Interrupt OB



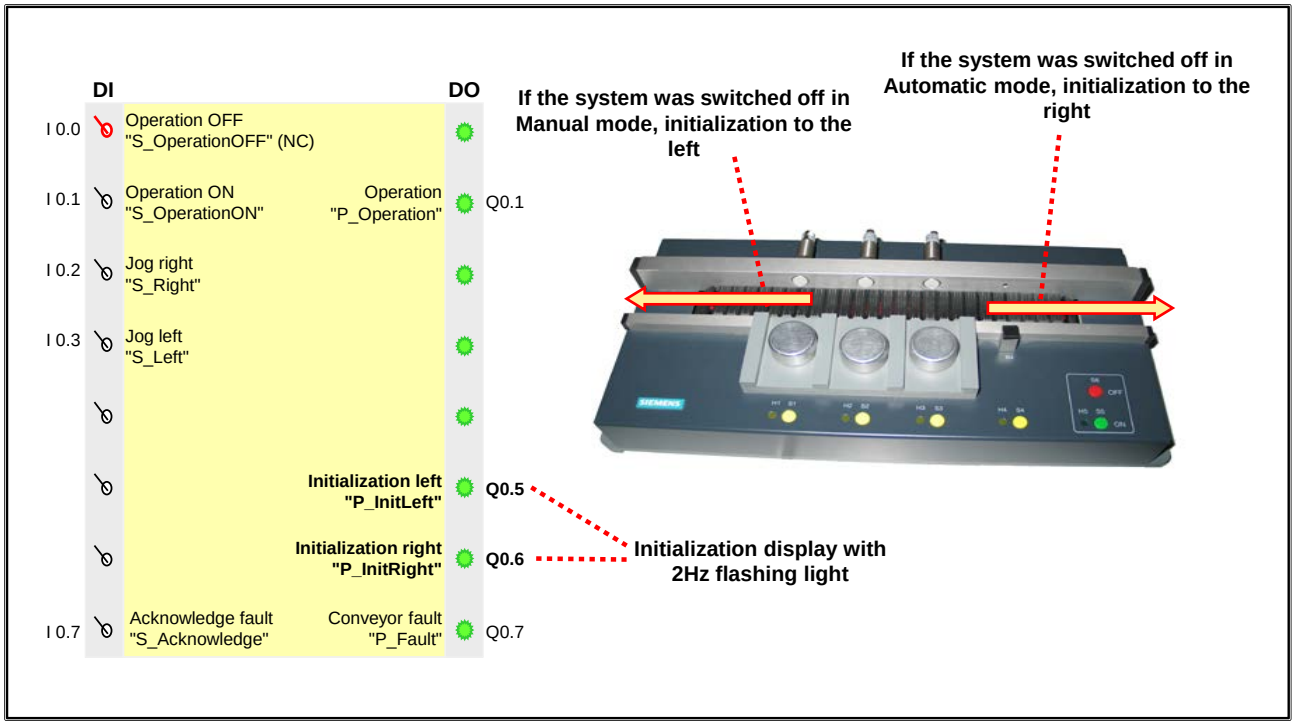
Task

The new variable "DB_Memory".Init_Right must be set in a Startup OB and reset in a Time-delay interrupt OB after 5 seconds.

What to Do

1. Insert the new organization blocks "OB_Startup_1" of the type Startup and "OB_Time delay interrupt_1" of the type Time-delay interrupt.
2. In the OB "OB_Startup_1", make an assignment to the variable "DB_Memory".Init_Right so that it is assigned the value TRUE.
3. In a further network, call the function "SRT_DINT" and declare it so that the OB "Time delay interrupt_1" is started after 5 seconds.
   Caution: The parameter OB_NR only accepts the OB number or the constant name of the OB not the symbol name.
4. In order to be able to supply the parameters SIGN and RET_VAL with actual parameters, declare the relevant temporary variables "#Startinfo" (WORD) and "#ret_V" (INT).
5. In "OB_Time delay interrupt_1", reset the variable "DB_Memory".Init_Right.
6. Save, compile and download your project.
7. Test the new function.

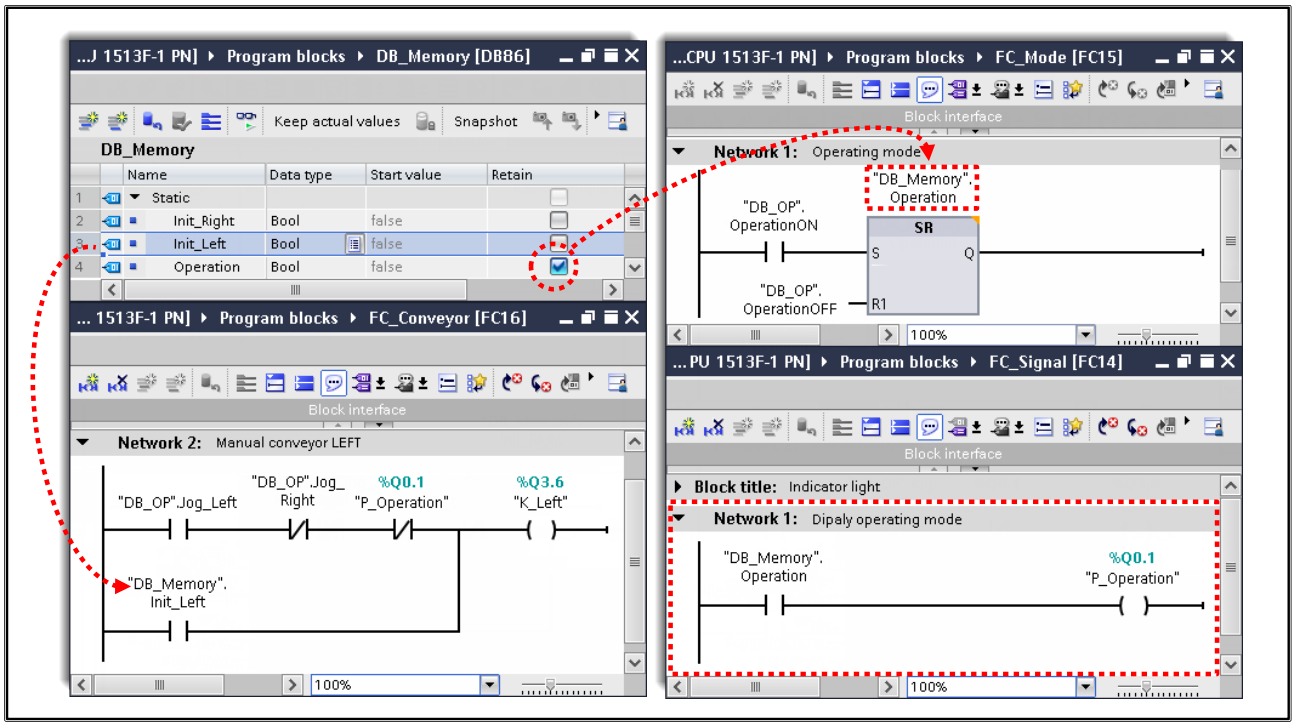
13.6. Additional Task Description



Task Description

When switching on the system, the initialization is to depend on the operating status at the time when the system was switched off. If the system had the operating status Operation ON (Automatic) when it was switched off, then the transport conveyor is to move to the right. If the system had the operating status Operation OFF (Manual), when it was switched off, then the initialization movement is to the left. In addition, the relevant initialization movement is signaled with a 2Hz flashing light at the LEDs Q0.5 "P_InitLeft" and Q0.6 "P_InitRight".
In order to achieve this, the operating mode must be stored as retentive.

13.6.1. Additional Exercise 3: Preparing for the Initialization Expansion



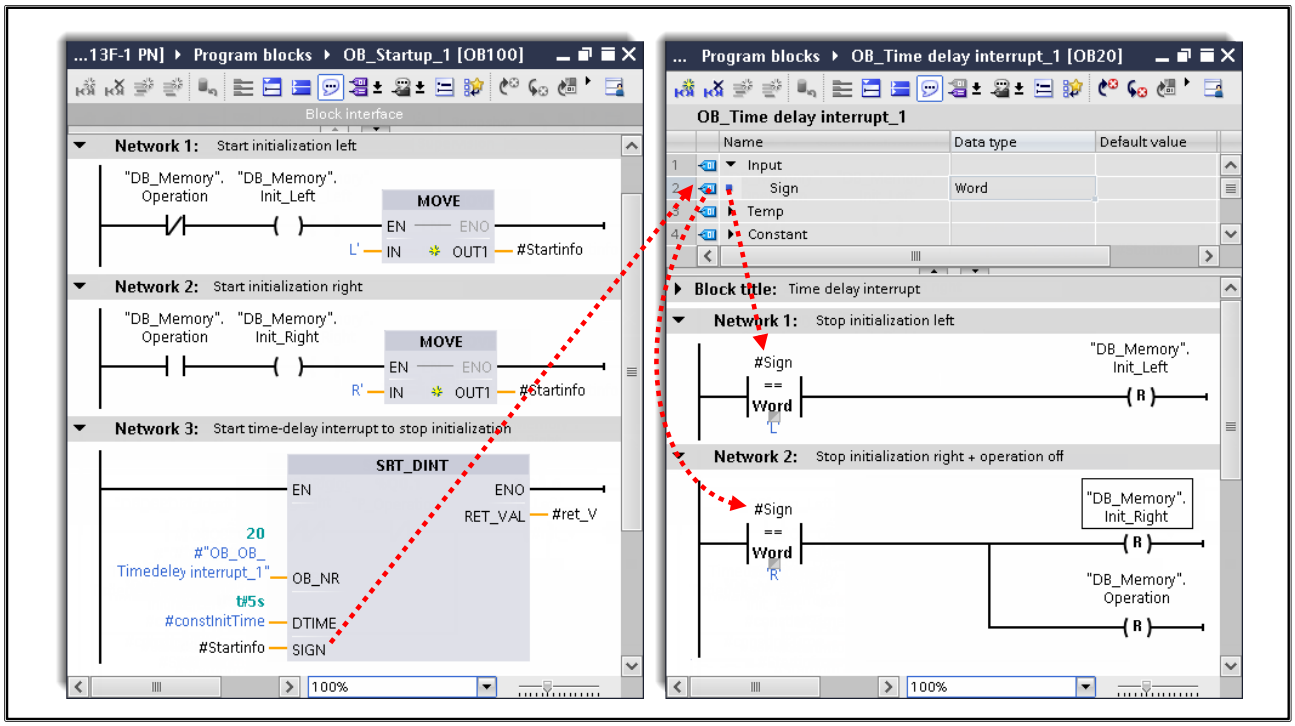
Task

You are to declare a variable for the movement to the left and assign it as a condition for a movement to the left. Furthermore, a retentive variable is required for the operating mode.

What to Do

1. In "DB_Memory", declare two new variables "Init_Left" and "Operation".
2. Give the variable "Operation" the property "Retentive".
3. In "FC_Conveyor", insert a new OR logic operation for the control of the output "K_Left" and as a further condition give the variable "DB_Memory".Init_Left a TRUE signal.
4. For the operating mode in "FC_Mode", set the variable "DB_Memory".Operation instead of the output "P_Operation".
5. To signalize the operating mode in "FC_Signal", assign the status of the variable "DB_Memory".Operation to the output "P_Operation".
6. Save the program modifications.

13.6.2. Additional Exercise 4: Initialization to the Left/Right



What to Do

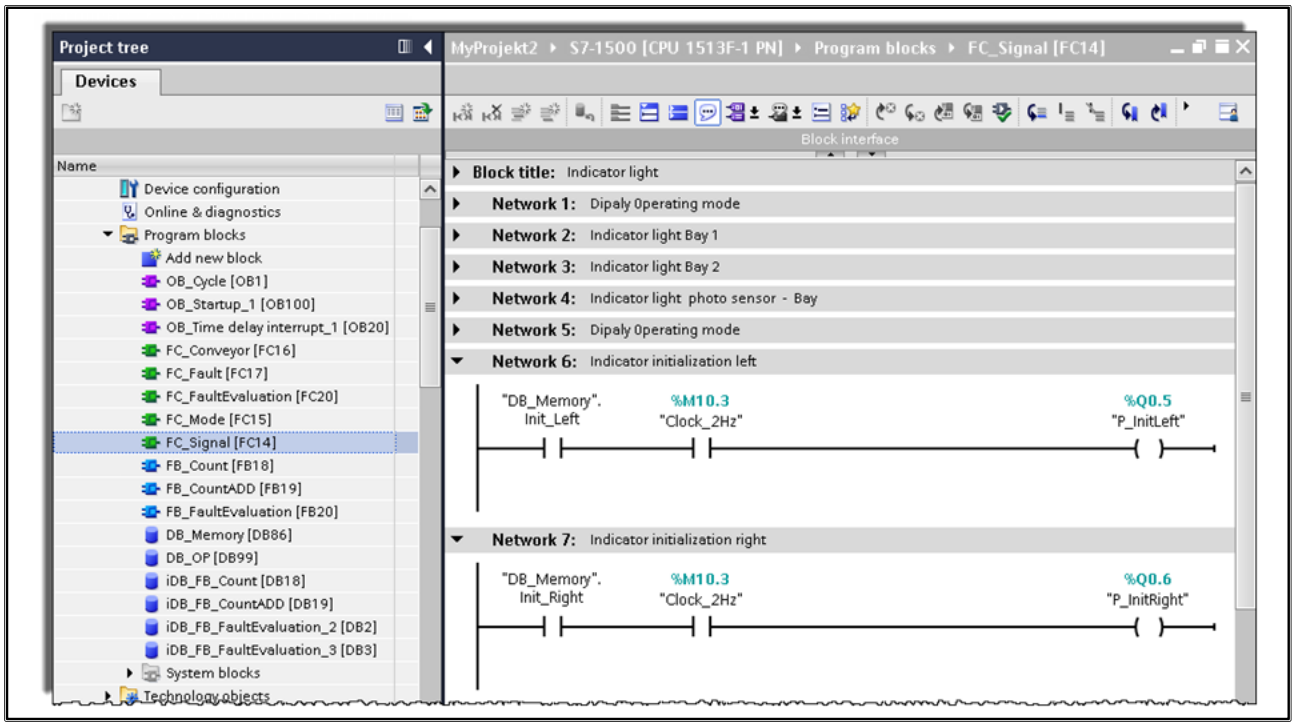
In the Startup OB, after evaluating the variable "DB_Memory".Operation, the variable "DB_Memory".Init_Right or "DB_Memory".Init_Left is assigned the value TRUE.

Furthermore, the character ‘L’ for left or ‘R’ for right is written in the temporary variable #Startinfo. The call of "OB_Time delay interrupt_1" remains unchanged. However, after evaluating the input parameter #Sign, the variables "DB_Memory".Operation and "DB_Memory".Init_Right or "DB_Memory".Init_Left are reset in this.

Solution Hints

1. In "OP_Startup_1", assign the variable "DB_Memory".Init_Left the value TRUE and write the CHAR value ‘L’ in the variable #Startinfo when the variable "DB_Memory".Operation has the status FALSE.
2. Change the assignment of the variable "DB_Memory".Init_Right so that it only gets the value TRUE when the variable "DB_Memory".Operation has the status TRUE. Additionally in this case, the CHAR value ‘R’ is written in the variable #Startinfo.
3. The call of the function "SRT_DINT" remains unchanged.
4. Reset the variable "DB_Memory".Init_Left in "OB_Time delay interrupt_1" when the CHAR value ‘L’ is passed with the input parameter #Sign.
5. If the CHAR value ‘R’ is passed with the parameter #Sign, then the variables "DB_Memory".Operation and "DB_Memory".Init_Right are reset.
6. Save the program modifications.

13.6.3. Additional Exercise 5: Displaying the Initialization



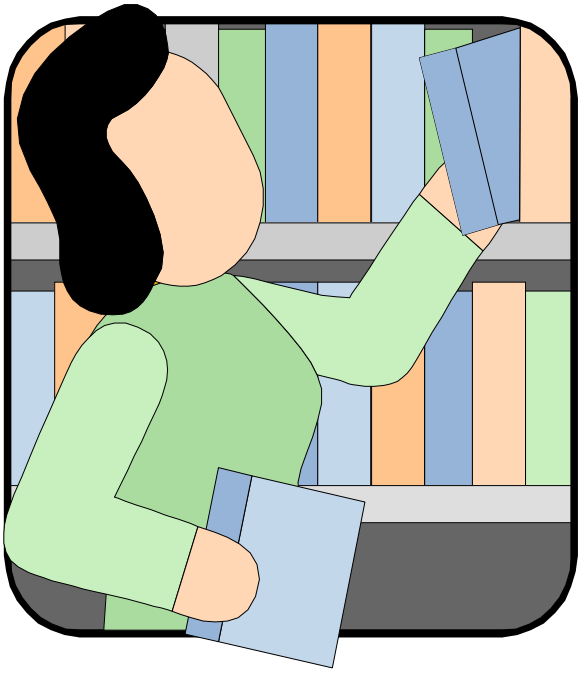
Task

The current initialization is to be displayed on the relevant LED "P_InitLeft" or "P_InitRight" with a 2Hz flashing light.

What to Do:

1. Open the function "FC_Signal" and insert two new networks.
2. With an AND instruction, interconnect the variable "DB_Memory".Init_Left and the clock memory "Clock_2Hz" and assign the result to the output Q0.5.
3. Give the output Q0.5 the symbol name "P_InitLeft".
4. Repeat the logic operation with the variable "DB_Memory".Init_Right and "Clock_2Hz" for the output Q0.6 and give it the symbol name "P_InitRight"
5. Save, compile and download the program modifications.
6. Test the new function.

13.7. Additional Information



13.7.1. S7-1200/1500: Global Error Handling with Asynchronous Error OBs

Type of Error	Example	Error OB	Priority
Time Error: Max. allowed cycle time exceeded once System reaction with OB: **RUN** without OB: **STOP** Max. allowed cycle time exceeded by more than double System reaction with OB: **STOP**	Exceeding the max. allowed cycle time, delayed call of a time OB	**OB 80**	**22**
Diagnostic Interrupt System reaction w/o OB: **RUN**	Wire break at diagnostics-capable module, power supply error	**OB 82**	**Can be set: 2..26**
Remove / Insert Interrupt System reaction w/o OB: **RUN**	Remove / Insert a module	**OB 83**	**Can be set: 2..26**
Rack Failure System reaction w/o OB: **RUN**	Failure of a DP-Slave or an IO-Device	**OB 86**	**Can be set: 2..26**

All events lead to an entry in the diagnostics buffer

Asynchronous Errors

Asynchronous errors occur asynchronous (independent) to the program execution and accordingly cannot be assigned to a defined program location.

Time Errors

They occur when the current cycle time exceeds the cycle monitoring time set in the Properties of the CPU.

Diagnostic Interrupts

They are triggered by diagnostics-capable modules, such as, analog modules in case of a fault (for example, wire break).

Remove/Insert Interrupts

These interrupts are triggered when modules are removed or inserted. When a module is inserted, the operating system checks whether the correct module type was used. With this function, it is possible to remove/insert modules while the system is running.

Rack Failure

A rack failure is detected with the failure of a rack, a subnet or a station of distributed I/O.

13.7.2. S7-1200/1500: Global Error Handling with Synchronous Error OBs

Type of Error	Example	OB	Priority
S7-1200 Programming error Access error System reaction w/o OB: RUN	Access to non-existing DB Direct access to non-existing or defective I/O module	**no OB exists**	※
Programming error System reaction w/o OB: STOP S7-1500	Access to non-existing DB	**OB 121 (only S7-1500)**	**Can be set: 2..26**
Access error System reaction w/o OB: RUN	Direct access to non-existing or defective I/O module	**OB 122 (only S7-1500)**	

All errors lead to an entry in the diagnostic buffer

Synchronous Errors

Synchronous errors occur synchronously (dependent) to the program execution and accordingly can be assigned to a defined program location.

With a programming error, OB121 is called; with an access error, OB122. If, in case of an error, the appropriate synchronous error OB does not exist in the CPU, the CPU switches to the STOP state.

S7-1500:

You can set the priority of the synchronous error OBs from 2 to 26. The register contents that the interrupted block has used are not available in the error OB and cannot be manipulated by means of system functions.

13.7.3. OB Priorities and System Reaction

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of OBs
Startup*	1	100, ≥ 123	Ignore	0 to 100
Cyclic program*	1	1, ≥ 123	Ignore	0 to 100
Time-of-day interrupt*	2 to 24 (2)	10 to 17, ≥ 123	not applicable	0 to 20
time-delay interrupt*	2 to 24 (3)	20 to 23, ≥ 123	not applicable	0 to 20
Cyclic interrupt*	2 to 24 (8 to 17, frequency dependent)	30 to 38, ≥ 123	not applicable	0 to 20
Hardware interrupt*	2 to 26 (18)	40 to 47, ≥ 123	Ignore	0 to 50
Status interrupt	2 to 24 (4)	55	Ignore	0 or 1
Update interrupt	2 to 24 (4)	56	Ignore	0 or 1
Manufacturer-specific or profile-specific interrupt	2 to 24 (4)	57	Ignore	0 or 1
Isochronous mode interrupt	16 to 26 (21)	61 to 64, ≥ 123	Ignore	0 to 2
Time error	22	80	Ignore	0 or 1
Maximum cycle time exceeded once			STOP	
Diagnostic error interrupt	2 to 26 (5)	82	Ignore	0 or 1
Removal/insertion of modules	2 to 26 (6)	83	Ignore	0 or 1
Rack error	2 to 26 (6)	86	Ignore	0 or 1
MC servo interrupt	17 to 26 (25)	91	not applicable	0 or 1
MC interpolator interrupt	16 to 26 (24)	92	not applicable	0 or 1
Programming error (only for global error handling)	2 to 26 (7)	121	STOP	0 or 1
I/O access error (only for global error handling)	2 to 26 (7)	122	Ignore	0 or 1

Varying amounts of OBs can be created for every OB type. Numbers that are smaller than 123 are permanently assigned to certain OBs and numbers that are larger / equal to 123 are freely selectable.