Table of contents

SFC Flags

SFC Flags

SFC flags are implicitly generated variables with predefined names. You can use them to influence the processing of an SFC diagram. You can use these flags, for example, to display timeouts or reset step chains. In addition, you can activate jogging mode specifically to activate transitions. You have to declare and activate these variables in order to have access to them.

SFC flags

Name	Data Type	Description
`SFCInit`	Bool	`TRUE`: PLC Engineering resets the sequence to the initial step. The other SFC flags are also reset (initialization). While the variable is `TRUE`, the initial step remains set (active), but its actions are not executed. Only when you set `SFCInit` again to `FALSE` is the POU further processed normally.
`SFCReset`	Bool	This function behaves similar to `SFCInit`. However, PLC Engineering continues processing after the initialization of the initial step. For example, in the initial step, you could immediately reset the `SFCReset` flag to `FALSE`.
`SFCError`	Bool	`TRUE` if a timeout occurs in an SFC diagram. If second timeout occurs in the program, it is not registered unless you previously reset the variable `SFCError`. The declaration of `SFCError` is a requirement for other flag variables to function for controlling the chronological sequence (“SFCErrorStep”, `SFCErrorPOU`, `SFCQuitError`).
`SFCEnableLimit`	Bool	Used specifically for activating (`TRUE`) and deactivating (`FALSE`) the timeout control in steps using `SFCError`. If you declare and activate this variable (SFC settings), then you must set it to `TRUE` for `SFCError` to work. If you do not, then the timeouts are ignored. The is useful, for example, at start-up or in manual operation. If you do not declare the variable, then `SFCError` will work automatically.

		The requirement is the declaration of `SFCError`.
`SFCErrorStep`	String	Stores the name of the step that caused a timeout, which was registered by `SFCError`. The name is kept until the registered step error is reset by means of `SFCQuitError` (FALSE -> TRUE). The requirement is the declaration of `SFCError`.
`SFCErrorPOU`	String	Stores the name of the block in which a timeout occurred and was registered by `SFCError`. The name is saved until the timeout is reset by `SFCQuitError`. The requirement is the declaration of `SFCError`.
`SFCQuitError`	Bool	As long as this Boolean variable is `TRUE`, PLC Engineering pauses the processing of the SFC diagram and any timeout, saved in the variable `SFCError`, is reset. If you reset the variable to `FALSE`, then all previous times in the active steps are reset. The requirement is the declaration of `SFCError`.
`SFCPause`	Bool	As long as this variable is `TRUE`, PLC Engineering pauses the processing of the SFC diagram.
`SFCTrans`	Bool	`TRUE` if a transition is active.
`SFCCurrentStep`	String	Shows the name of the active step, regardless of the time monitoring. In parallel branches, the name of the step of the rightmost branch line is always stored.
`SFCTip`, `SFCTipMode`	Bool	Controls the jogging mode of the SFC block. If you enable this flag with `SFCTipMode=TRUE`, then you can activate the next step only by setting `SFCTip` to `TRUE`. While `SFCTipMode` is set to `FALSE`, transitions can also be used to continue activation.
`SFCErrorAnalyzation`,		Contains as string all variables that contribute to the total value `TRUE` of `SFCError` (timeout in one step). `SFCError` needs to be activated for this. `SFCErrorAnalyzation` implicitly uses the function of the POU `AnalyzeExpression` of the library `Analyzation`.
`SFCErrorAnalyzationTable`,		

Contains in a table all variables that contribute to the total value `TRUE` of `SFCError` (timeout in one step). `SFCError` needs to be activated for this.

`SFCErrorAnalyzationTable` implicitly uses the function of the POU `AnalyzeExpressionTable` of the library `Analyzation`.

Implicit generation of SFC flags

PLC Engineering declares SFC flags automatically when you activate the respective options. You can set this option in the “SFC Settings” tab of the properties dialog for each POU, or in the “SFC” project settings dialog for each SFC POU in the project.

The SFC settings for the SFC flags of individual POUs are effective only if you have not selected the “Use defaults” option. When you select this option, the settings apply that were defined in the project settings.

SFC flags that you declare in the SFC settings dialog are visible only in the online view of the SFC block.

See also

- ↘ “Flag”

Explicit generation of SFC flags

Manual declaration, which was necessary in CoDeSys V2.3, is now only required to enable write access from another block. In this case, you should note that when you declare the flag in a global variable list, you must deactivate its “Declare” setting in the SFC settings dialog. If you do not do this, then a local SFC flag is implicitly declared that PLC Engineering uses instead of the global variable.

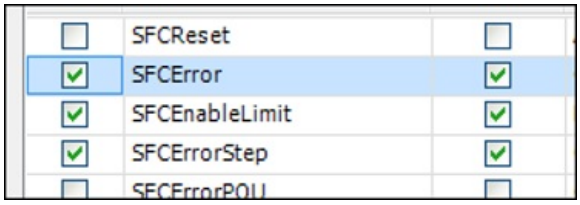
Application example for SFCError

Example

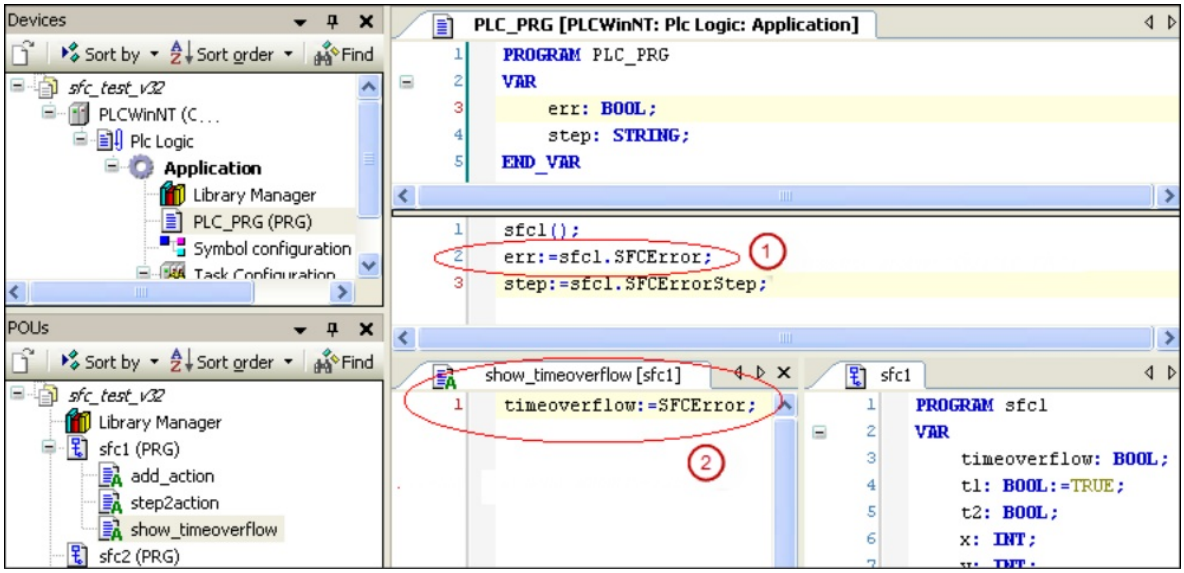
You have created an SFC block named `sfc1`, which contains the `s1` step. You have defined timeouts in the step properties. (See "Online view of SFC block `sfc1`" below.)

If for any reason the `s1` step remains active longer than its time properties have permitted (timeout), then PLC Engineering sets the `SFCError` flag to permit access by the application.

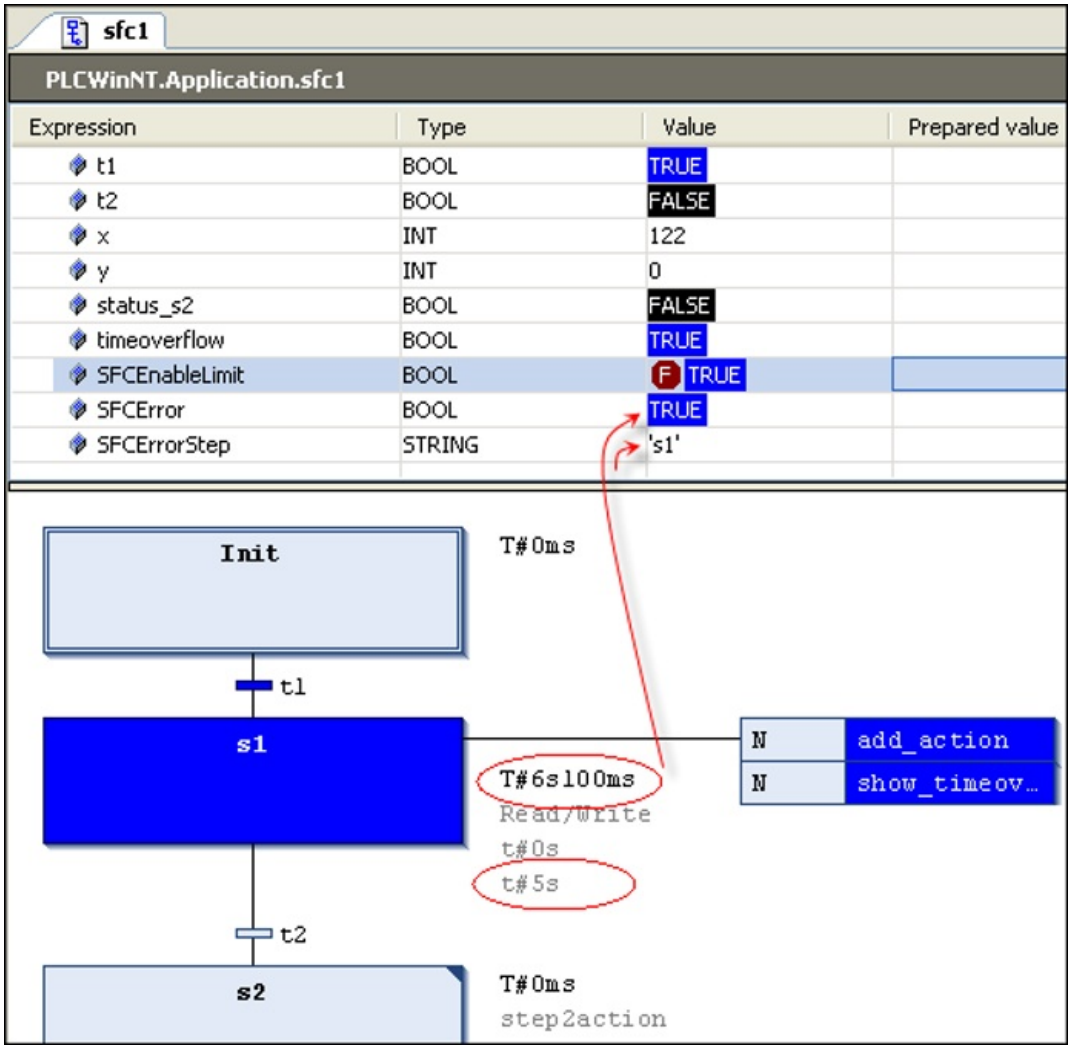
To permit access, you have to declare and activate the SFC flag in the SFC settings. If you have only declared it, then the SFC flag is only displayed in the online view of `sfc1` in the declaration part, but it has no function.



Now the SFC flag can be referenced within the POU, for example in an action (2) or outside of the block (1).



Online view of the SFC block `sfc1`



`SFCError` is TRUE as soon as a timeout occurs within `sfc2`.

Note that you can use the flags `SFCErrorAnalyzation` and `SFCErrorAnalyzationTable` to determine the components of the expression that contributes to the value TRUE of the `SFCError`.

See also

- ↘ “Library "Analyzation"”

Access to the flags

Syntax for access:

You assign the flag directly within the POU: `<variable name>:=<SFC flag>`

Example

```
checkerror:=SFCerror;
```

From another POU with POU name: `<variable name>:=<POU name>.<SFC flag>`

Example:

```
checkerror:=SFC_prog.SFCerror;
```

If you need write access from another block, then you also have to declare the SFC flag explicitly as a VAR_INPUT variable in the SFC block or globally in a GVL.

Example

Local declaration:

```
PROGRAM SFC_prog
VAR_INPUT
  SFCinit:BOOL;
END_VAR
```

Global declaration in a global variable list:

```
VAR_GLOBAL
  SFCinit:BOOL;
END_VAR
```

```
PROGRAM PLC_PRG
VAR
  setinit: BOOL;
END_VAR
SFC_prog.SFCinit:=setinit;  // write access to SFCinit in SFC_prog
```

See also

- ↘ “Library "Analyzation"”