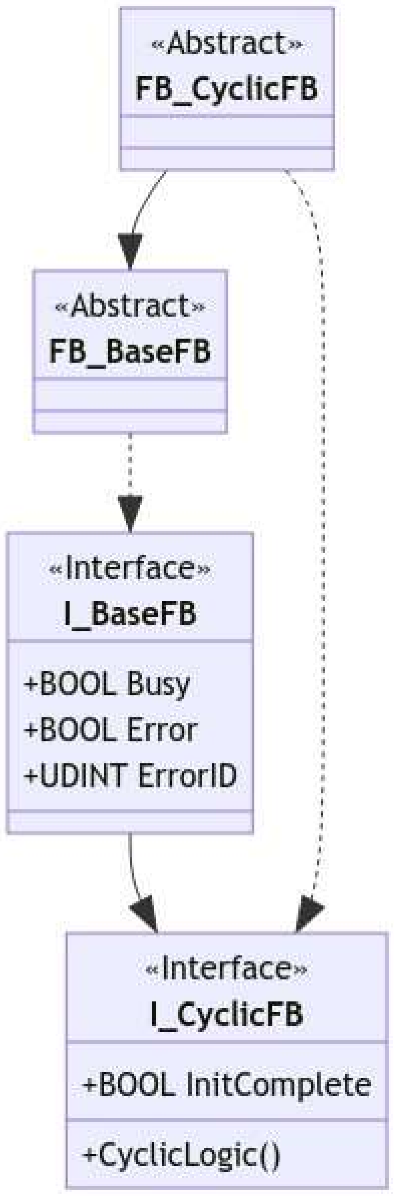
SPT Base Types

Last updated by | Nick Higgins | Sep 27, 2022 at 9:24 AM EDT

Contents

Class Diagram



Interfaces

I_BaseFB

Defines the most basic functionality of any function block used within the framework

Property	Type	Access	Description
Busy	BOOL	RO	Function block is performing some action
Error	BOOL	RO	Function block has encountered an error condition
ErrorID	UDINT	RO	Error-specific identifier

I_CyclicFB

(Extends I_BaseFB)

Adds to `I_BaseFB` the concept of initialization, as well as a unified entry point for cyclical code to be called. You may have collections of `I_CyclicFB` which are iterated through, calling `CyclicLogic()` on each (see SPT PackML Base).

Property	Type	Access	Description
InitComplete	BOOL	RO	Flag indicating that the function block is ready to use

Method	Return Type	Access	Description
CyclicLogic	null	PUBLIC	Entry point for code execution

Function Blocks

FB_BaseFB

(abstract, implements `I_BaseFB`)

Contains property backers for all `I_BaseFB` properties. This is the most basic building block of all framework function blocks. Can be directly inherited--if so, entry point is up to the developer.

FB_CyclicFB

(abstract, extends `FB_BaseFB` , implements `I_CyclicFB`)

Contains property backers for all `I_CyclicFB` properties. `CyclicLogic()` is introduced as the entry point--**no code should be written in the body of function blocks extending** `FB_CyclicFB`

Design Notes

Throughout the framework libraries a common pattern is used for initialization routines and how they are called.

Most function blocks will implement `I_CyclicFB` by way of inheriting `FB_CyclicFB` . The entry point for these function blocks is `CyclicLogic()` . `FB_CyclicFB` already contains a local variable backing the `InitComplete` property: `_InitComplete : BOOL` . We can utilize this in our function blocks and assure all necessary initialization steps have been carried out before executing any further code. This can be useful, for example, to make sure pointers are initialized before referencing them. Another example may be waiting for another function block to set a property on our function block--useful when implementing the Observer pattern.

```
IF NOT _InitComplete THEN
    _InitComplete := Initialize();
    RETURN;
END_IF

...code to run once initialization is complete
```