LamaPLC: Simatic TIA Portal know-how

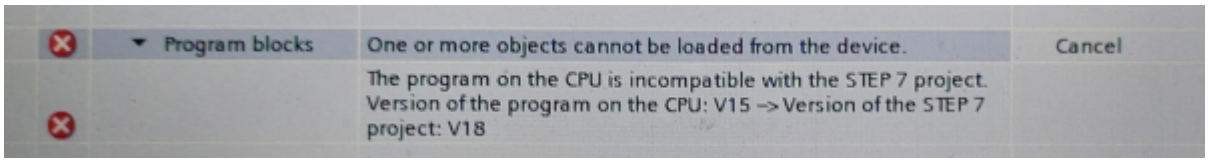
I predict that in the chapter, I will write the descriptions in my own words, as if I were trying to explain things to a friend. The description may not be professional this way, but my goal is for anyone who reads to understand what is described.

I have quite a lot of experience (more than 20 years) with Simatic systems, based on which I plan to develop this site.

If you have a specific question or request, please let me know! > Impressum

Management of TIA Portals with different versions

As a development engineer, I have to work with several different versions of TIA-Portal at the same time. I test the latest (currently 20), and develop new projects in 18. And I need to be able to open the old projects somehow. In theory, several TIA versions can exist on the machine at the same time, but in the case of WinCC (especially in the case of WinCC professional) this is not true. On my machine, only the current version always runs *"directly"*, and the rest are available on a virtual machine. Apparently, the V18 running on the current machine cannot open the older V15, so when I try to connect to the CPU online, I get the following error message:



In this case, I have two options:

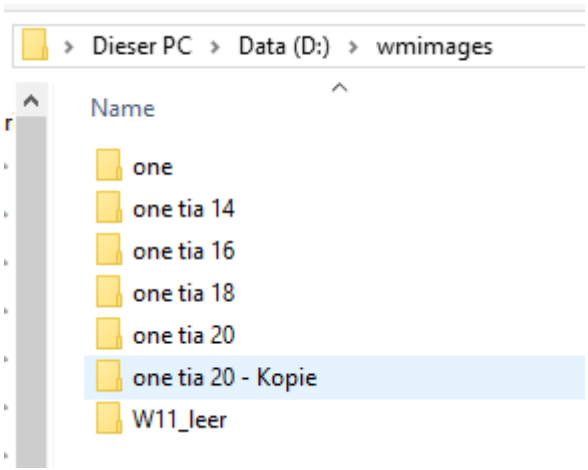
1.) Delete the old V15 project from the CPU and overwrite it with the current V18 version. This solution can be implemented if:

- I'm pretty sure the version ported on PC (V15→V18) is DEFINITELY the same as the CPU version
- I have the opportunity to test the new version locally on the CPU
- I have a save from the old (V15) CPU version

2.) First connect ON-LINE with the PC to the old (V15) TIA Portal, save it and modify / repair it.

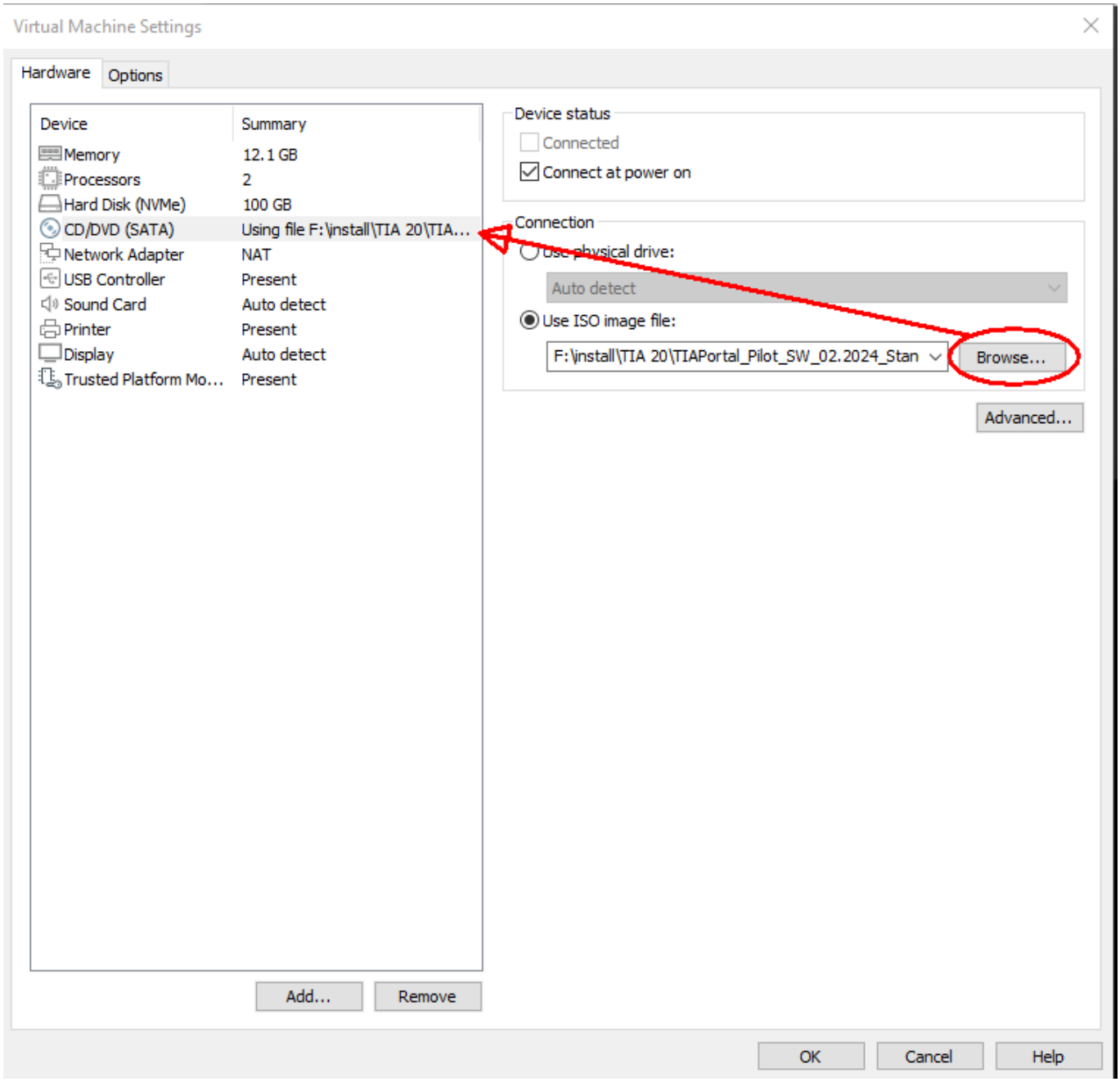
Second, if I can test the new version on the CPU, then download the ported V18 version to the CPU. In many cases, the focus is on the possibility of testing, and that's where the possibility of porting fails. "**Don't touch a working system!**" - this is usually the basic principle, so the old (V15) version remains on the CPU even after possible modifications and downloads. In this case, however, it is good to have several versions of TIA Portal on your PC that can be used simultaneously.

The use of a virtual machine is a big help in this. I use **VmWare**, a program available at a relatively normal price, its operation is very stable. I have saved several virtual machines simultaneously from various TIA portals, I always open the one that can *"talk"* to the CPU.

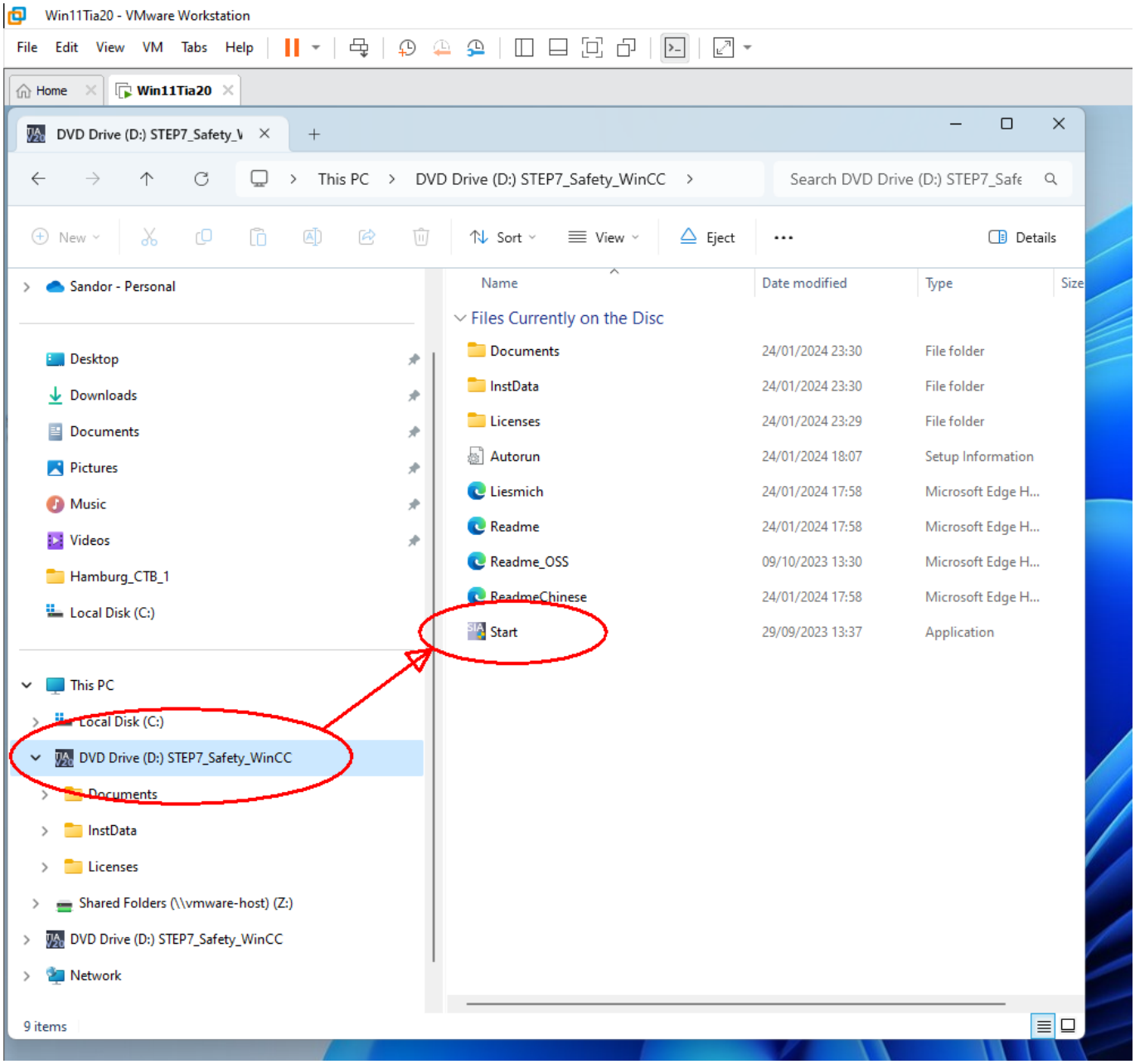


I always have a current *“empty”* Windows machine (*“one”* in the picture above). I install the current TIA version on it. For example, in the case of V15, type “tia 15 download” into Google. this will give you the download path, in this case this.

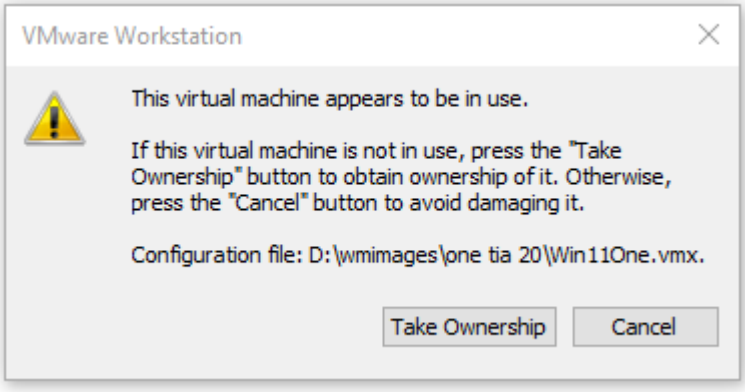
On the virtual machine, look for CD/DVD (SATA) in the settings and set the path to the downloaded TIA Portal ISO file here:



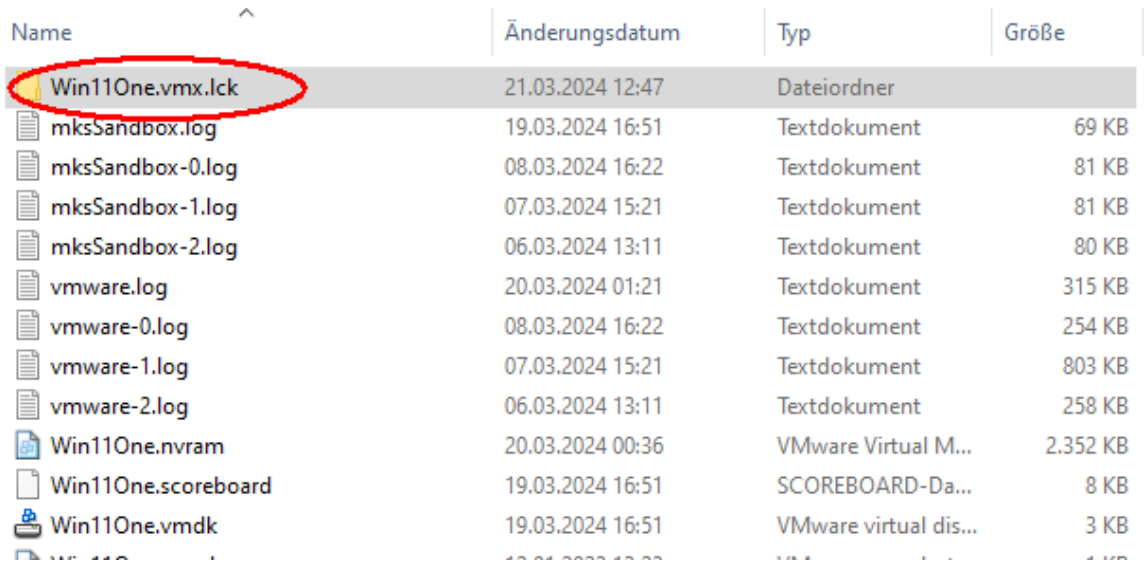
When we open the VmWare image, the DVD drive automatically opens the ISO file and we can start the installation:



Important: If the virtual machine is not shut down properly (this often happens to me), its lock "remains" *and does not allow you to open the machine again, for example:*



In this case, the locking directory must be deleted in the directory of the given virtual machine:



Name	Änderungsdatum	Typ	Größe
Win11One.vmx.lck	21.03.2024 12:47	Dateiordner	
mksSandbox.log	19.03.2024 16:51	Textdokument	69 KB
mksSandbox-0.log	08.03.2024 16:22	Textdokument	81 KB
mksSandbox-1.log	07.03.2024 15:21	Textdokument	81 KB
mksSandbox-2.log	06.03.2024 13:11	Textdokument	80 KB
vmware.log	20.03.2024 01:21	Textdokument	315 KB
vmware-0.log	08.03.2024 16:22	Textdokument	254 KB
vmware-1.log	07.03.2024 15:21	Textdokument	803 KB
vmware-2.log	06.03.2024 13:11	Textdokument	258 KB
Win11One.nvram	20.03.2024 00:36	VMware Virtual M...	2.352 KB
Win11One.scoreboard	19.03.2024 16:51	SCOREBOARD-Da...	8 KB
Win11One.vmdk	19.03.2024 16:51	VMware virtual dis...	3 KB

And, tadaaam, you can open the VmWare image.

FC / FB description

FC: function

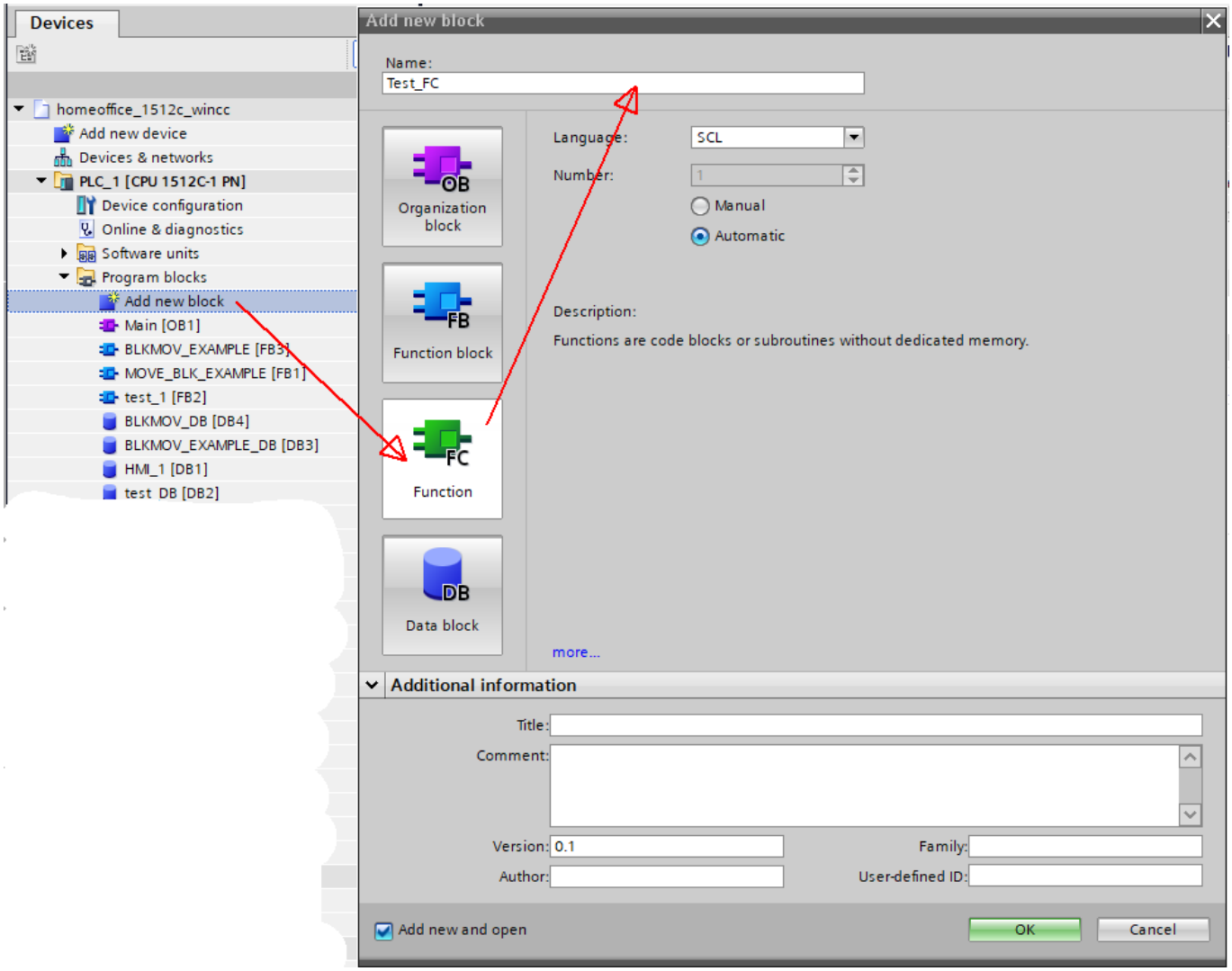
FCs are programming blocks that do not require static variables, that is, they do not need internal information that would be stored from one cycle to another. It is therefore not necessary to assign data blocks, i.e. DBs, to FCs.

In the references, I denote FCs as follows: _FC_

for example: _FC_ Result := **ABS** (Value);

I repeat because it is important: The internal variables of the FC are valid only within one cycle, they lose their values for the next cycle. If you need to store information, use the FB or call DB, merkel.

Add new FC



- double click to “Add new block”
- chose “Function”
- type a new name

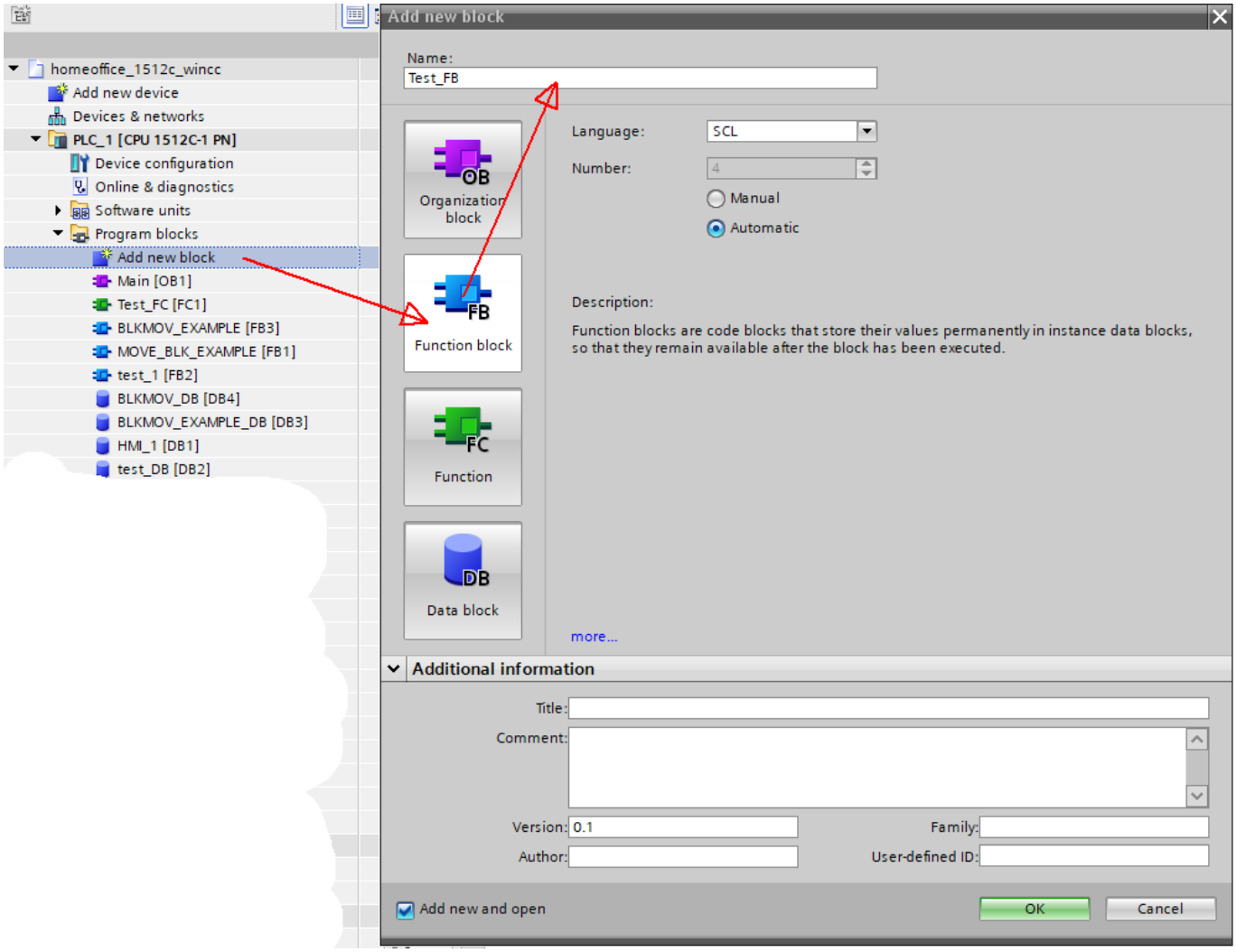
FB: function blocks

FBs are a bit more complex modules than FCs. FBs are able to define cyclical variables, these are “static” variables. These are not actually stored in the FB, but will be assigned a datablock (DB) to each FB and it will store these variables. These DBs should be called IDBs (instant datablocks) because, unlike global DBs, they are assigned to a specific FB. In addition to storing variables, IDBs are also suitable for storing embedded FBs, this will be discussed later. The IDB will be assigned to a given SC during the first call.

In the references, I denote FBs as follows: _FB_

for example: _FB_ RetVal (INT):= BLKMOV (SRCBLK := source block (VARIANT); DSTBLK ⇒ destination block (VARIANT); ENO ⇒ operation enable);

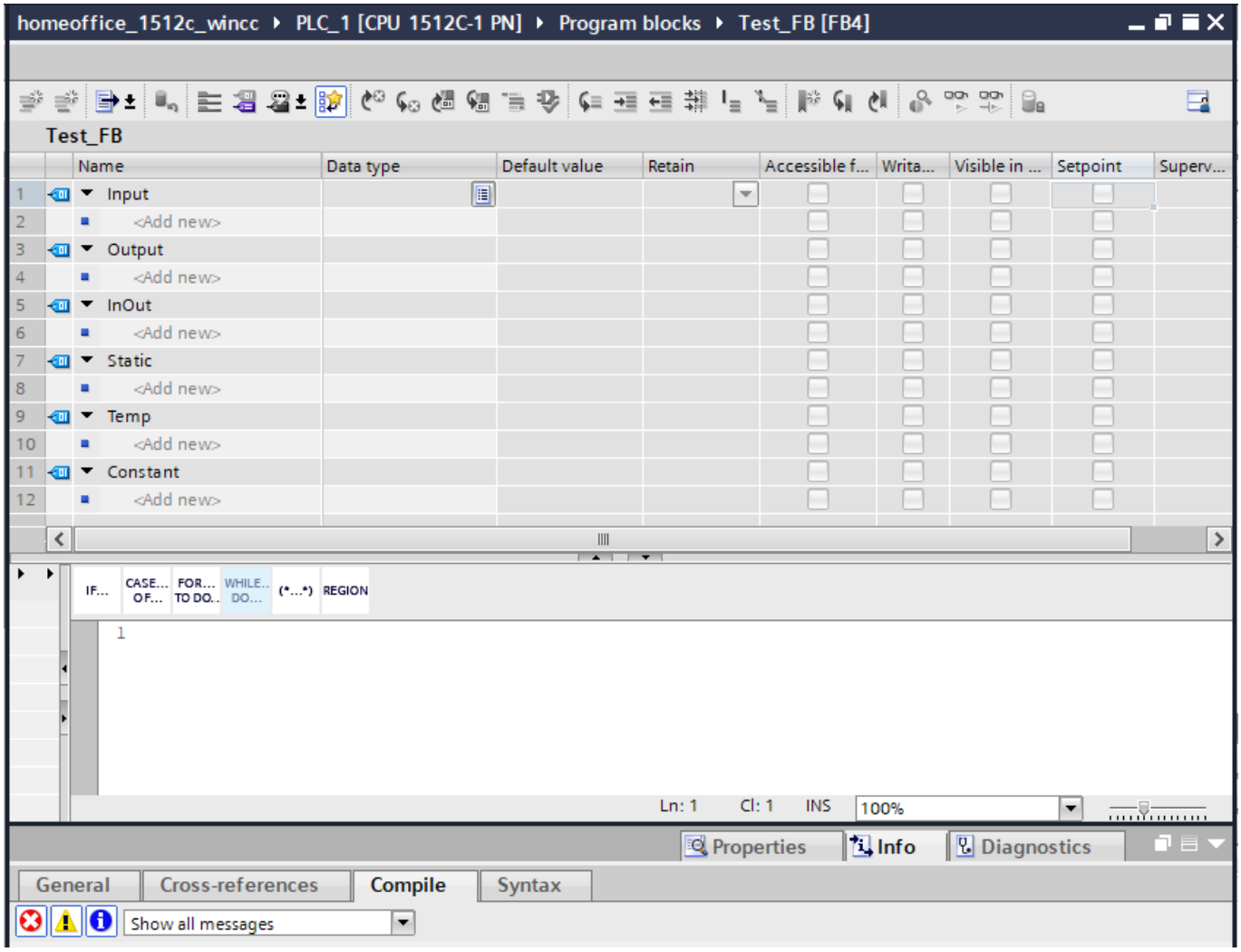
Add new FB



- double click to “Add new block”
- chose “Function block”
- type a new name

FC / FB variable types

When we open the newly created Fc or FB (FB in the example) we will see the following image:



Field reports:

Name	**FC / FB**	**description**
Input	FC and FB	Variables read by modules.
Output	FC and FB	Variables write by modules.
InOut	FC and FB	Variables read and then written back by modules. Changes made to the module are reflected in the outputs.
Static	FB only	static variables: their value is retained because they are stored in the IDB assigned to the FB. If you open the IDB, you can follow the change in the value of the variables there.
Temp	FC and FB	Temporary variables: they can only be used within the given cycle, they will lose their value at the time of another call.
Constant	FC and FB	Constant values: their value must not change.
Return	FC only	the return value of the function, which is the name of the function. In the case of an external call: retval: = Function_name ();

Call FCs

Calling FCs is easier:

1. the part of the program (FC, FB or OB) from which the FC is to be called must be opened.
2. drag-and-drop the FC from the **“instructions”** window, or simply plug it in. For example, after

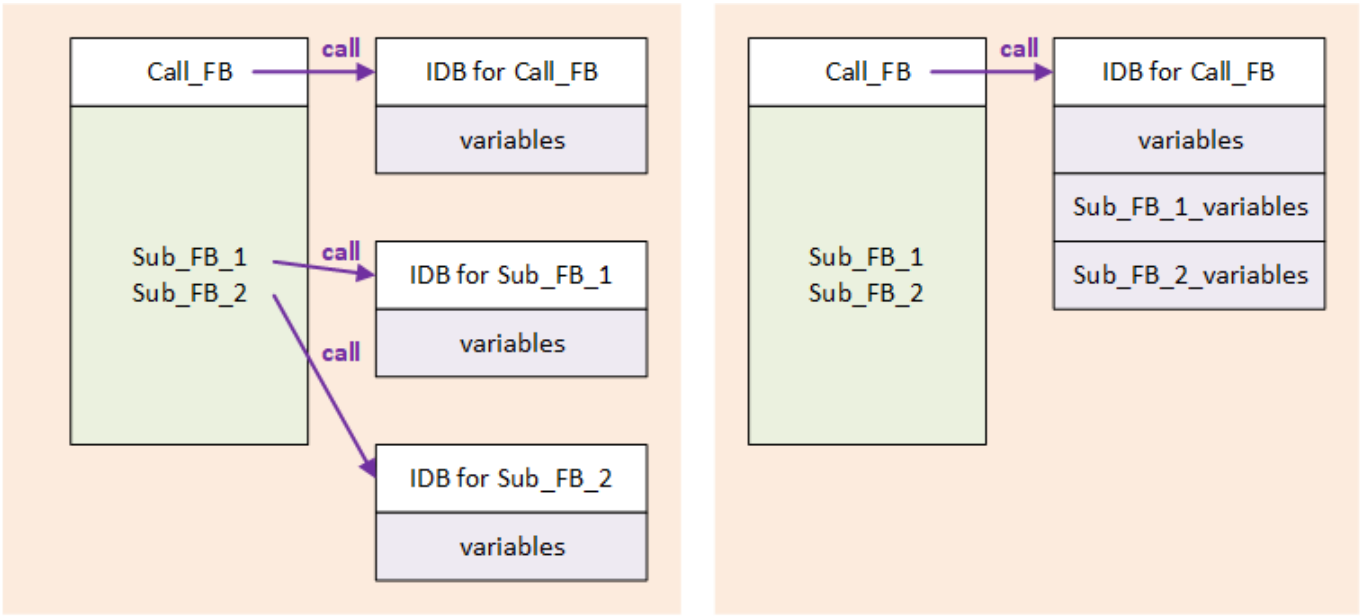
indentation, “LN” FC looks like this: *LN (_real_in_)*

3. you need to know, for example from here that the “LN” FC is calling floating-point numbers, ie REAL or LREAL.
4. the return value must also be given, eg: out:= LN(in); where in and out are REAL.

Call FBs

When calling FBs, we have two options:

1. each invited sub_FB receives its own IDB
2. sub_FBs are treated as an embedded function and their data area will be “taped” to the IDB of the calling FB.



each sub_FB calls its own IDB

the data of the sub_FBs is embedded in the IDB of the call_DB

https://www.ob121.com/doku.php?id=de:s7:tia_knowhow

Let's see this through examples, let's take one of the most commonly used methods, TON:

First method, call with separate IDBs:

```
3
4  "IEC_Timer_0_DB".TON(IN := #start,
5                       PT := t#12s);
6
7  "IEC_Timer_0_DB_1".TON(IN:=#start,
8                         PT:=t#12s);
9
10
11
```

		#start	FALSE
		#start	FALSE

Using two TON FBs, both calling their own IDBs

Static PT Time T#0ms T#0MS ET Time T#0ms T#0MS IN Bool false FALSE Q Bool false FALSE	Content of one of the IDBs

Second method, call embedded FBs:

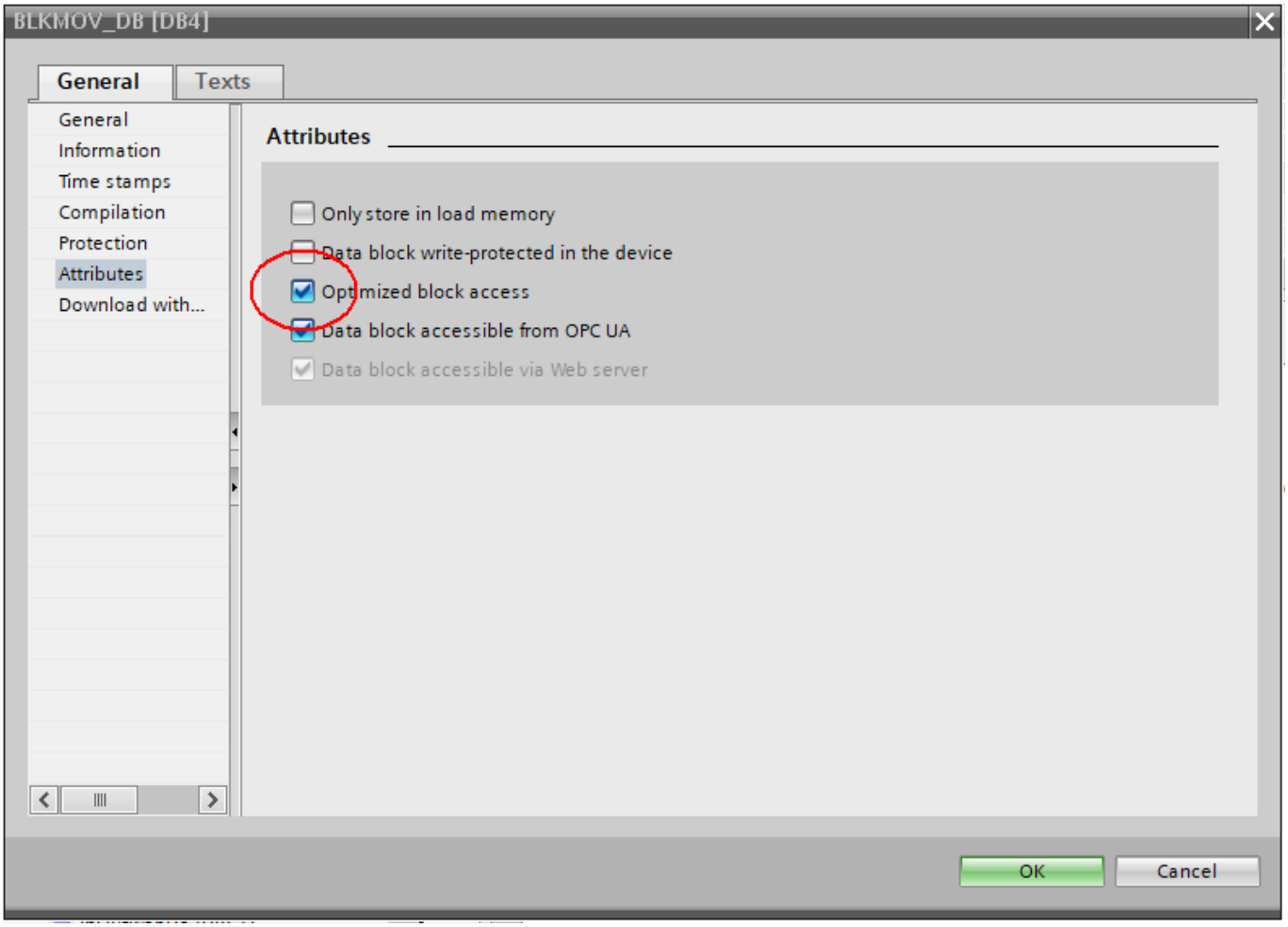
Static time_1 TON_TIME Non-retain time_2 TON_TIME Non-retain Temp start Bool <Add new> Constant No condition defined. IF... CASE... OF... FOR... TO DO.. WHILE... DO... (*...*) REGION 1 2 #time_1(IN:=#start, 3 PT:=t#12s); 4 5 #time_2(IN := #start, 6 PT := t#24s); 7 8 #start FALSE #start FALSE	In the case of embedding, the functions must be called from the “static” block of the calling FB (just type “TON” for the “data type”. FBs must be called in the program in much the same way, of course the DB call must be omitted.
test_3_DB Name Data type Start value Monitor value Re Input Output InOut Static time_1 TON_TIME PT Time T#0ms T#0MS ET Time T#0ms T#0MS IN Bool false FALSE Q Bool false FALSE time_2 TON_TIME PT Time T#0ms T#0MS ET Time T#0ms T#0MS IN Bool false FALSE Q Bool false FALSE	The image shows the data of the embedded FBs in the IDB of the calling FB. The great advantage of this method is that the program is easy to import / export / copy, as you only need one FB definition.

Switch off "optimization" attribute of DBs

By default, the TIA portal turns **on** the *“optimization”* attribute of DBs, which means that the order of variables in DBs is automatically optimized.

In many cases, this causes serious problems when the given variables have to remain in predefined positions, such as OPC, or typically any similar communication (Modbus, PN, ..) DB. For example, the BLKMOV command does not work with optimized DBs either.

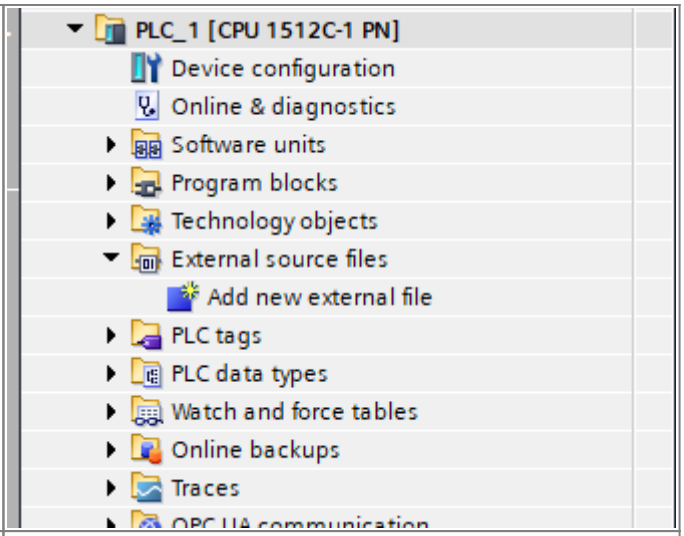
To deactivate, right-click on the DB and select **Properties...** and then **Attributes**:



The **“optimized block access”** attribute in the menu must be turned **off**.

For generated DBs, the *{S7_Optimized_Access: = 'TRUE'}* entry should be modified as follows: **{S7_Optimized_Access: = 'FALSE'}**.

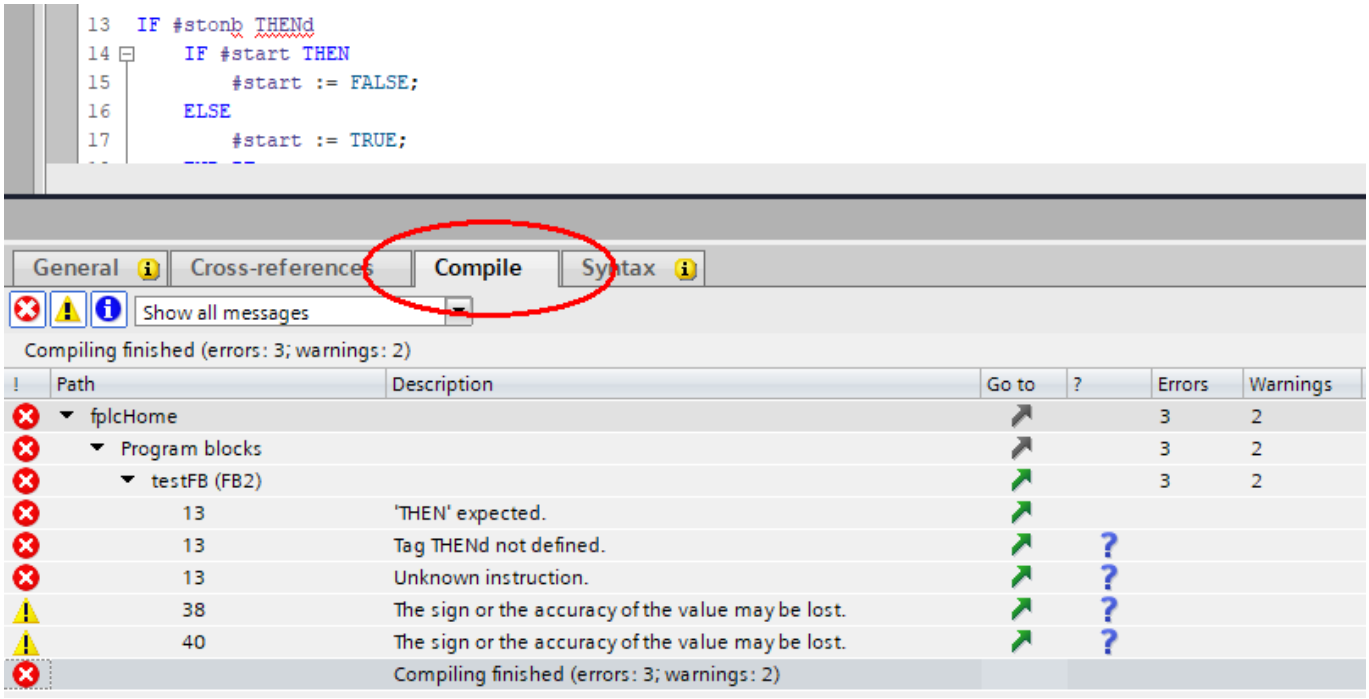
Import source code to the TIA portal

1: Double click to: “**Add new external file**” by “**External source files**” menu 2: Choose files and open	
3: right click to the file 4: choose “**Generate blocks from file**” 5: IF the message “**blocks can be overwriten**” comming: press OK. 6: Enjoy the code!	

Troubleshooting in the TIA Portal

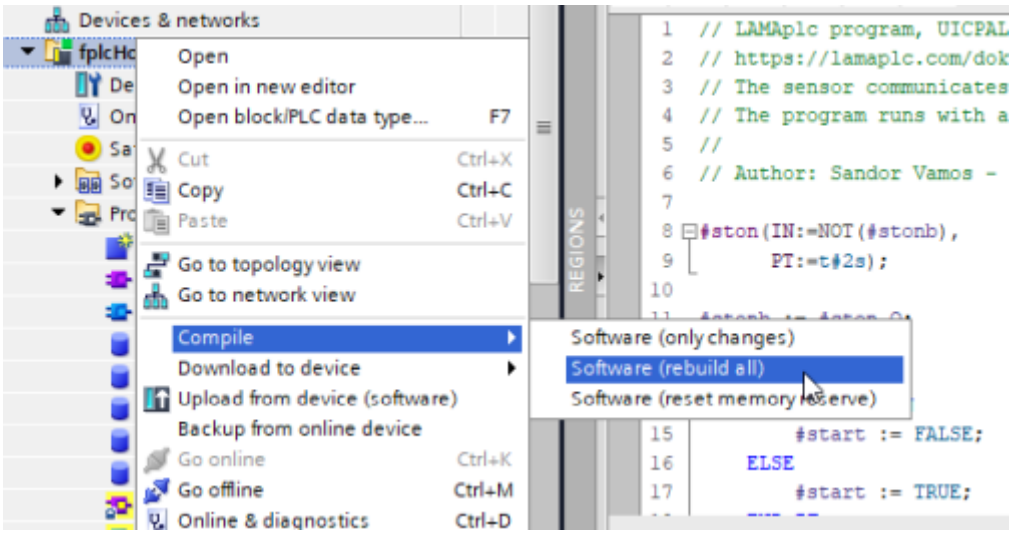
This is perhaps the “favorite” pastime of programmers. Errors are divided into Systematic and Semantic errors. Systematic are consistently repeated but formally correct errors that the compiler cannot *“protect”*, the programmer must find their cause.

Semantic error is simpler, the compiler also *“notices”* this. It is typically displayed as follows:  . The TIA-portal compiere also indicates warnings, as follows:  These do not prevent the compilation and download of the program, they only draw attention to possible causes of errors, for example data conversions where data may be lost. A typical error report on the TIA portal looks like this:



In the event of an error, the program automatically opens the *"Compile"* status window to review the errors. The location and presumed cause of the error/errors is indicated here. In this case, a typing error is indicated, and the location of the error in the code is indicated by the compiler with a red underline. At the bottom of the list, the program also displays warnings for some conversions, but these are only warning signals, they do not hinder the translation and download.

There are cases when the compiler cannot reveal the location or cause of the error, or the program throws a "strange" error message (I will copy this here the next time I encounter it). This error mainly occurs when you switch between online and offline status several times and make changes to the program in the process. In such cases, it is easiest to recompile the entire program and perform a full download:



Right click on the PLC → compile → Software (rebuild all). Caution, since this means a complete download, the PLC must normally be switched to STOP (and then restarted) for this operation.

simatic, s7, tia portal, fc, function, fb, function block, switch off db optimization, troubleshooting

This page has been accessed for: Today: 3, Until now: 5