Studio360

powered by

Smartware Studio™

Function Block Editor User's Guide

Version 5.0
February 2018

Smartware Technologies
1000 Young Street, Suite 450
Tonawanda, NY 14150

Sales and Support
(716) 213-2222

www.smartwaretech.com

Table of Contents

1. Introduction / About This Guide

Smartware Studio (also known as Studio360) is a versatile environment for use in all aspects of project development, engineering, estimating and document storage. This Guide will discuss the advanced EcoStruxure Function Block Editor available as part of the Software Development module.

For complete information about Smartware Studio, please refer to the following guides installed with the software and available from our web site at www.smartwaretech.com.

- *Smartware Studio Setup and Administration Guide*
- *Smartware Studio User's Guide*

2. Setup and Configuration

The Function Block Editor is installed as part of Smartware Studio and Studio360. To enable it, there are a few items to configure.

Requirements

The Function Block Editor requires:

- Smartware Studio / Studio360 Version 4.1.530 or later
- Visio 2010 or later (for now)
    - Visio 2013 or later is much preferred. Conversions of large Function Block programs will take several minutes in Visio 2010, but only seconds in later versions where the .vsdm file format can be used. We expect to eventually discontinue support for Visio 2010 in favor of Visio 2013 or later.
    - We are currently testing Visio 2019.
- If editing from within EcoStruxure, you must use WorkStation 1.9 or 2.0 or later.
    - Function Block programs can be edited entirely within Studio, and do not require EcoStruxure WorkStation for editing. WorkStation is required for simulation and testing.
    - We are currently testing Workstation 3.0.

Install the Latest Version of Studio

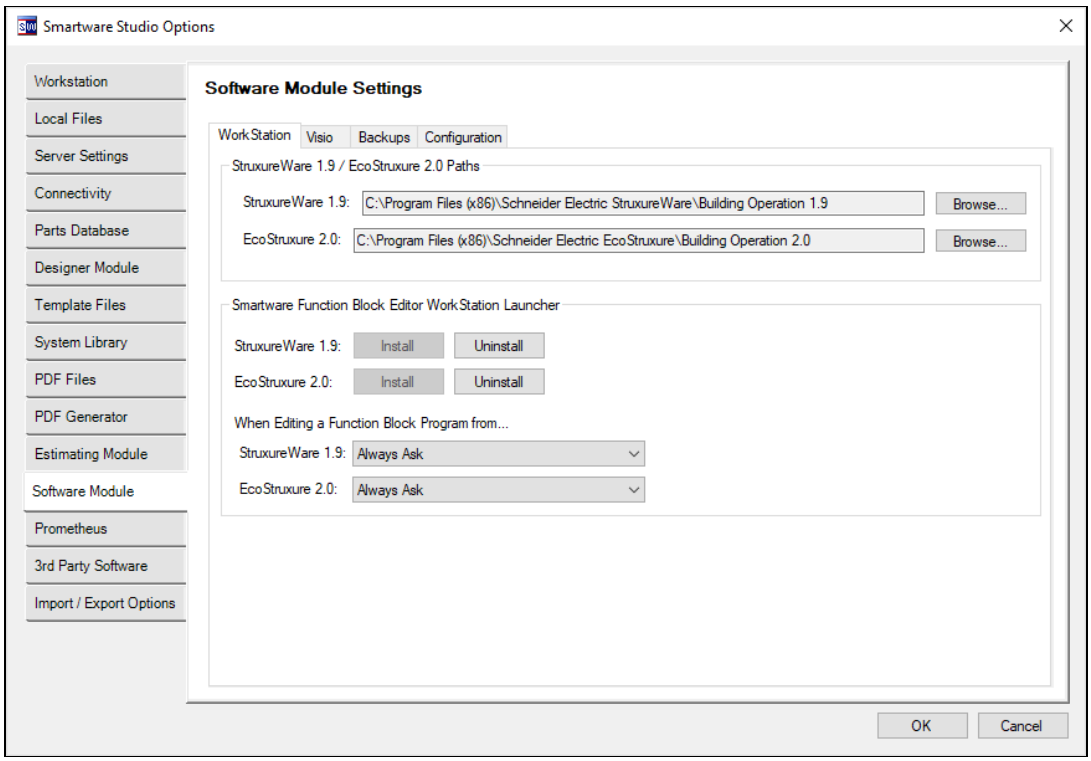
The Function Block Editor requires Version 4.0.530 or later. You can get the latest version (and beta versions) from our web site at www.smartwaretech.com on the Smartware Studio/Studio360 page.

Obtain a License for the Function Block Editor

Your workstation's license will need to be updated to enable this feature. Licenses can be purchased directly from Smartware Technologies.

Configure the Smartware Function Block Editor

Once installed, go to TOOLS→OPTIONS and select the *Software Module* tab



- There are separate settings for versions 1.9 and 2.0. You can use both versions on the same machine (assuming both versions of Workstation are installed and licensed).

EcoStruxure Install Folders

If you want to edit Function Block programs directly from EcoStruxure WorkStation, browse to find and specify the folder where it is installed.

- The default location on the C: drive should already be specified. If it is installed on a different drive, you will need to locate the folder.

Smartware Function Block Editor WorkStation Launcher

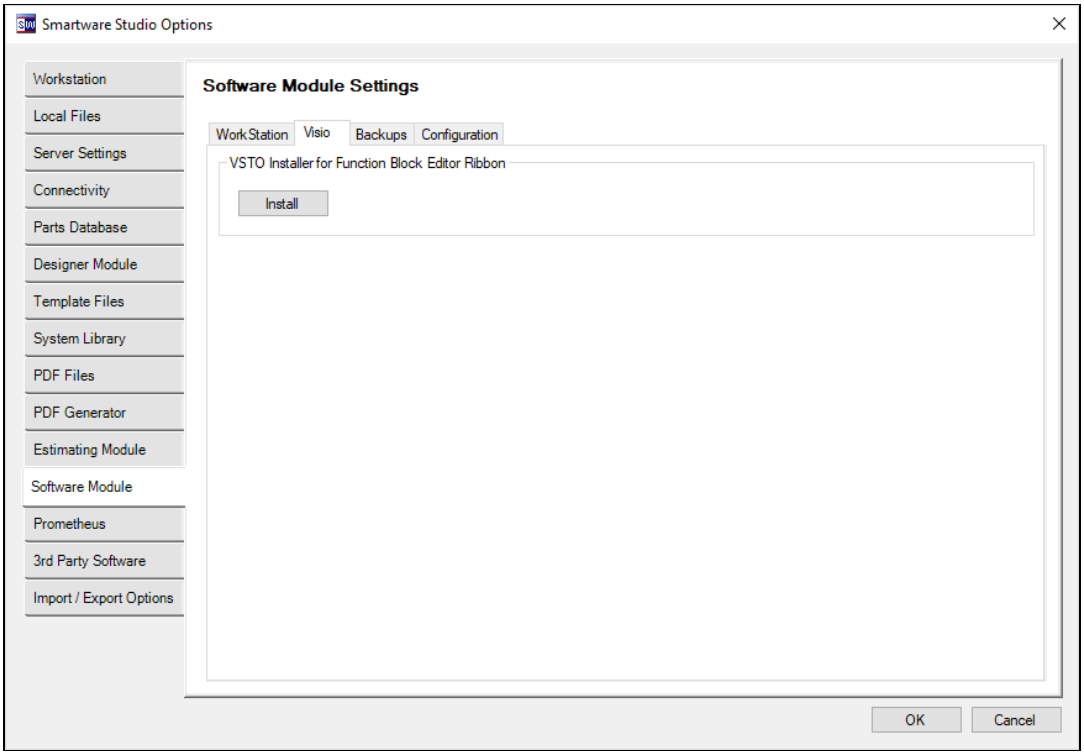
If using EcoStruxure WorkStation, click the INSTALL button to install the hooks for EcoStruxure to launch the Studio version of the Function Block editor.

- By default, the launcher will prompt to ask every time you edit a Function Block program in WorkBench whether to use the standard Function Block Editor or the Studio Function Block Editor. You can specify a default choice here.

- From time to time with a new release of Studio you may need to click UNINSTALL and then INSTALL to update the launcher.
- If you connect via HTTPS, you will need to have installed a permanently trusted certificate. Instructions for this should appear when you log into Workstation.

VSTO Installer for Function Block Ribbon

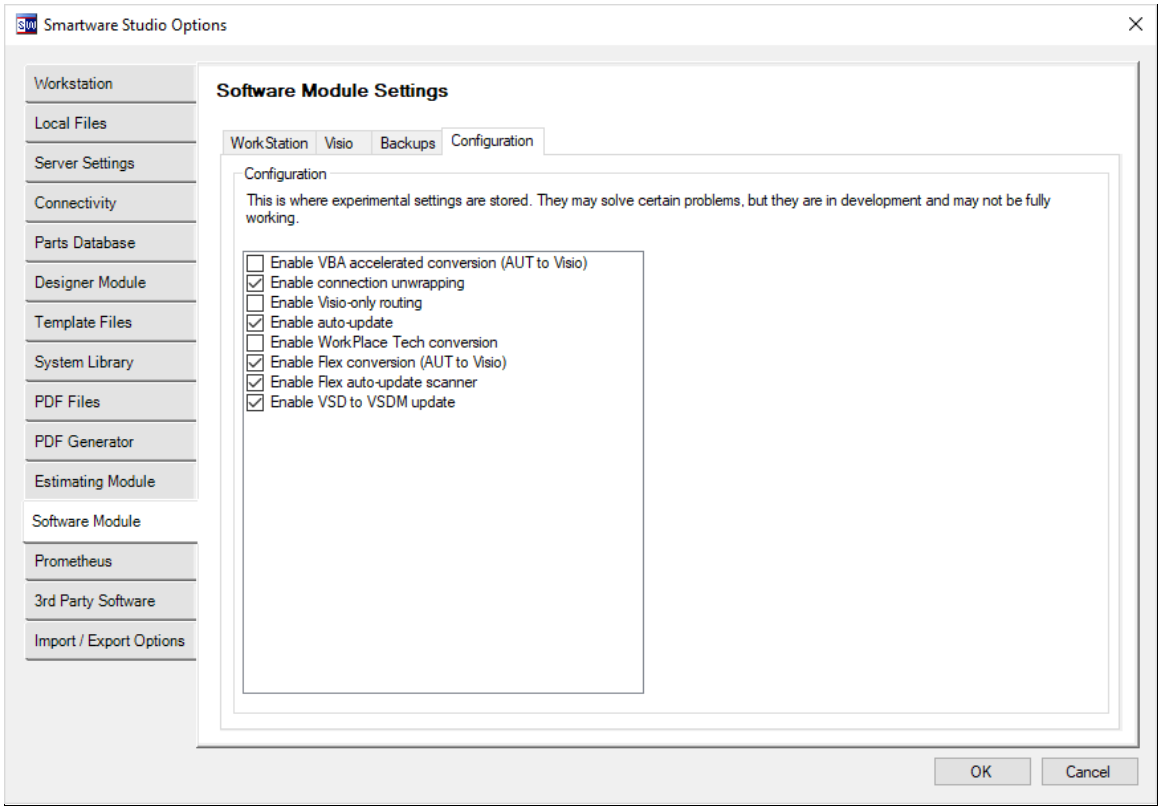
Studio will add and enable a new Function Block ribbon in Visio when editing a program. Depending on the version of Visio and Microsoft Office installed on your workstation, you may need to install this Microsoft update to make it work properly.



- It is likely that Office 2010 users will need to install this, but there are other cases when it is necessary.
- You can wait to see if the ribbon works without it before running this installer.

Additional Features and Options

As different features become available, they may not be fully stable yet, or some people may not want to use these newer features yet. The *Configuration* tab is for enabling or disabling these features.

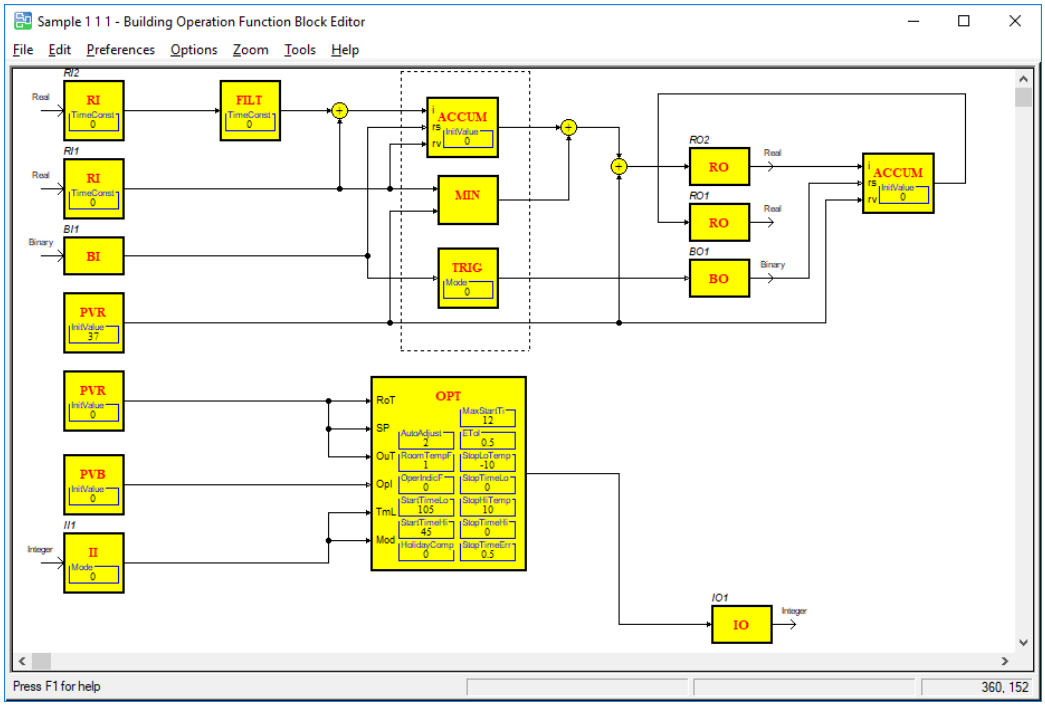


3. Editing a Function Block Program in Studio using Visio

The Studio Function Block editor allows you to edit your programs in our Visio environment, both offline and directly from an Automation Server or Enterprise Server.

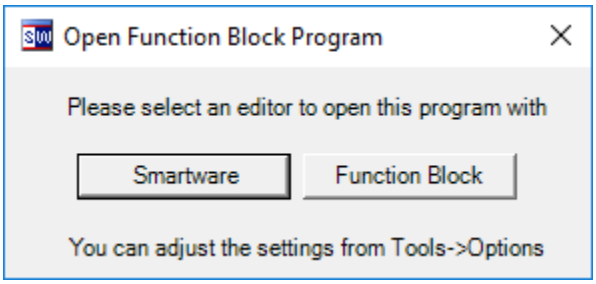
Editing a Function Block Program from EcoStruxure

Function Block programs are generally stored in the EcoStruxure station on a Function Block Program node. You can then right-click on the node and select EDIT to edit the program using the standard EBO Function Block Editor.



- In the EBO editor you can select FILE→EXPORT and FILE→IMPORT to save and load the program as an .AUT file.

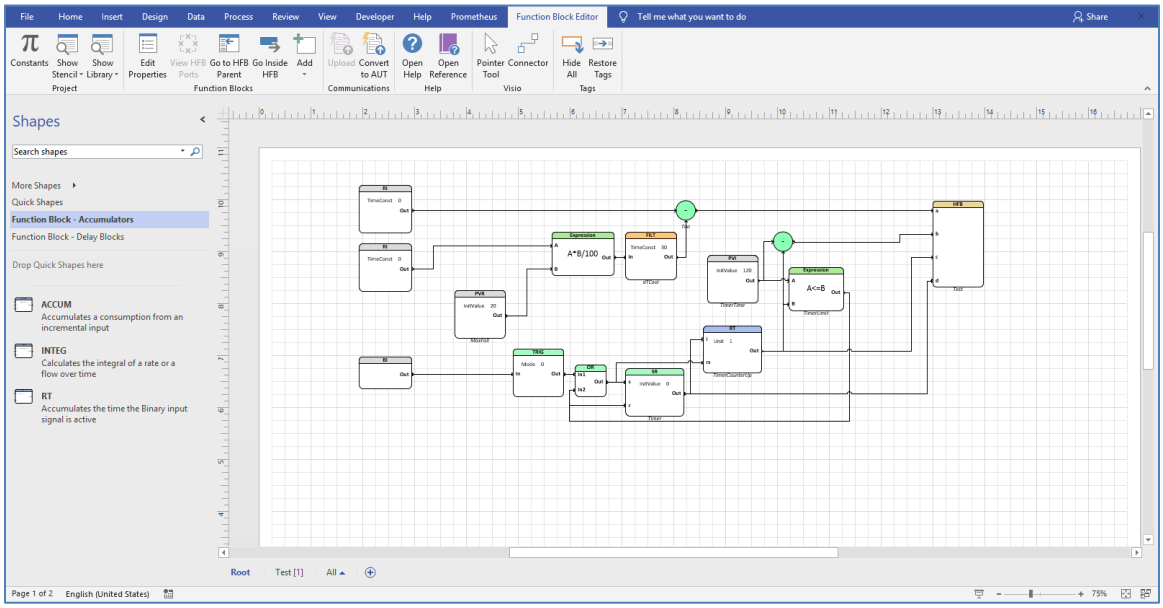
When Studio's Function Block Editor is installed, when you select EDIT you will be given a choice of which editor to use:



- Choose SMARTWARE to use the Studio Visio-based editor
- Choose FUNCTION BLOCK to use the standard EBO Function Block Editor
- To set it to choose one editor as the default without prompting, go to TOOLS→OPTIONS and the *Software Module* tab.

When the program is opened in the Smartware editor in Visio:

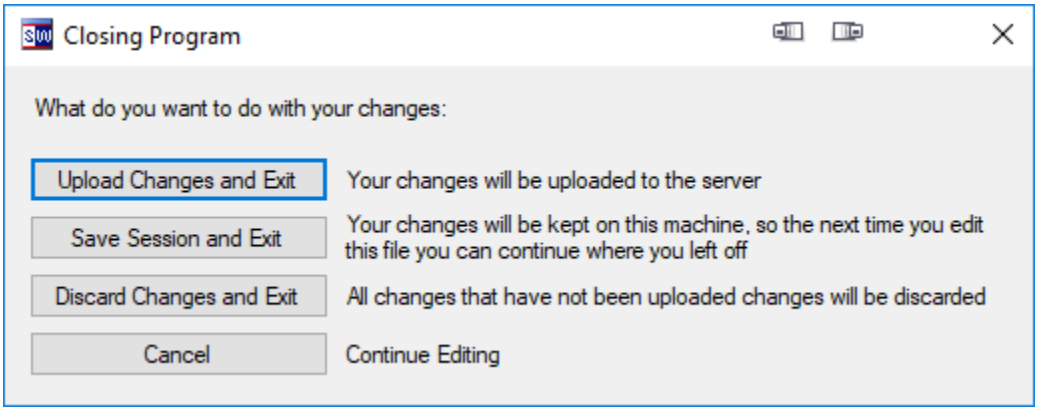
- The .AUT file is downloaded from the station through EBO WorkBench
- The program is converted to the Smartware Visio format and saved as a Visio drawing file (.vsd)
- It is then opened in Visio with the Studio Function Block Ribbon enabled.



Once opened, you can create and edit the program using the Studio Visio blocks, stencil and tools, described later in this chapter.

- To convert the Visio file back to an AUT file and upload it to the station, click the UPLOAD button in the ribbon. The Graphics Viewer view of the program in WorkBench should update within a few seconds.

When you exit Visio, you can choose how to save:



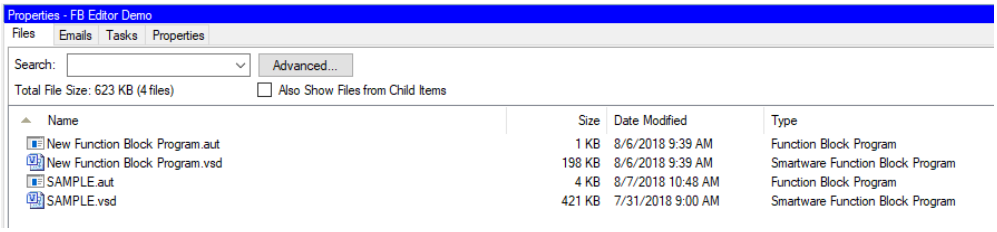
- If you choose SAVE SESSION AND EXIT, the file will not be uploaded to the server. Instead the Visio version will be saved on your computer so you can continue editing it later.
    - The next time you launch that file from WorkStation, you will be prompted to continue your previous work.
    - If you instead choose to load the .AUT from the station, your temporary version will be overwritten (but will still be available amongst the backup files).

Creating and Editing Programs without EcoStruxure

Studio does not need EcoStruxure to be installed on the workstation in order to edit Function Block programs. You can store .AUT files and Studio Function Block Visio files anywhere in your Studio projects (on a Network Tree device for an AS or an MP-X controller, for example) and convert and edit them from there.

- To create a blank .AUT file, right-click in the *Files* tab and select NEW→FUNCTION BLOCK PROGRAM.
- To edit an existing Function Block program from elsewhere, export it to an .AUT file from the EBO Function Block Editor (FILE→EXPORT) and copy it to the *Files* tab of a Studio project node.

When you double-click or OPEN an .AUT file, Studio will automatically convert it to a Smartware Function Block Visio drawing using the same name.



Studio will then open it in Visio with the Function Block Ribbon enabled.

Once converted, you can open the Visio version directly at any time to continue editing your program.

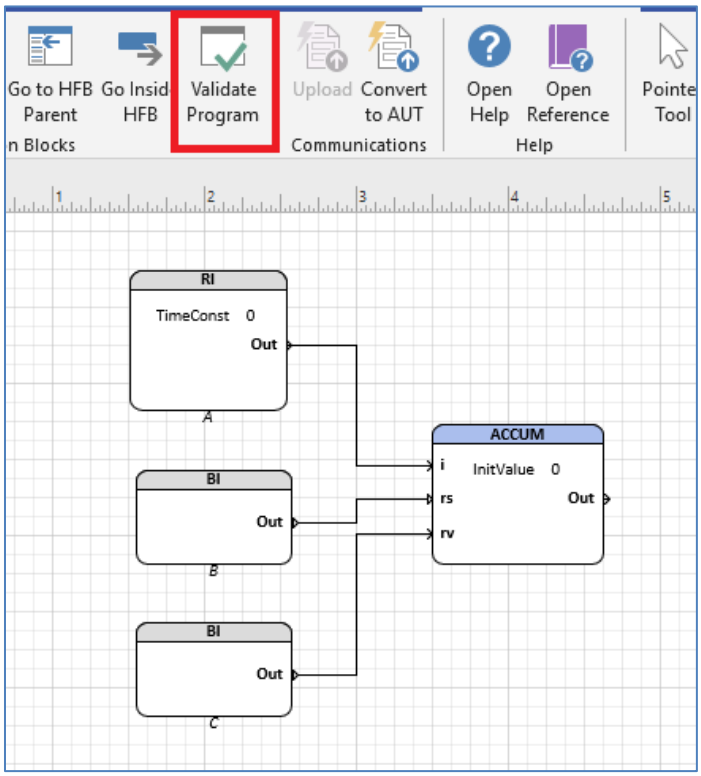
- The Function Block Editor ribbon is only shown when editing a program through Studio.

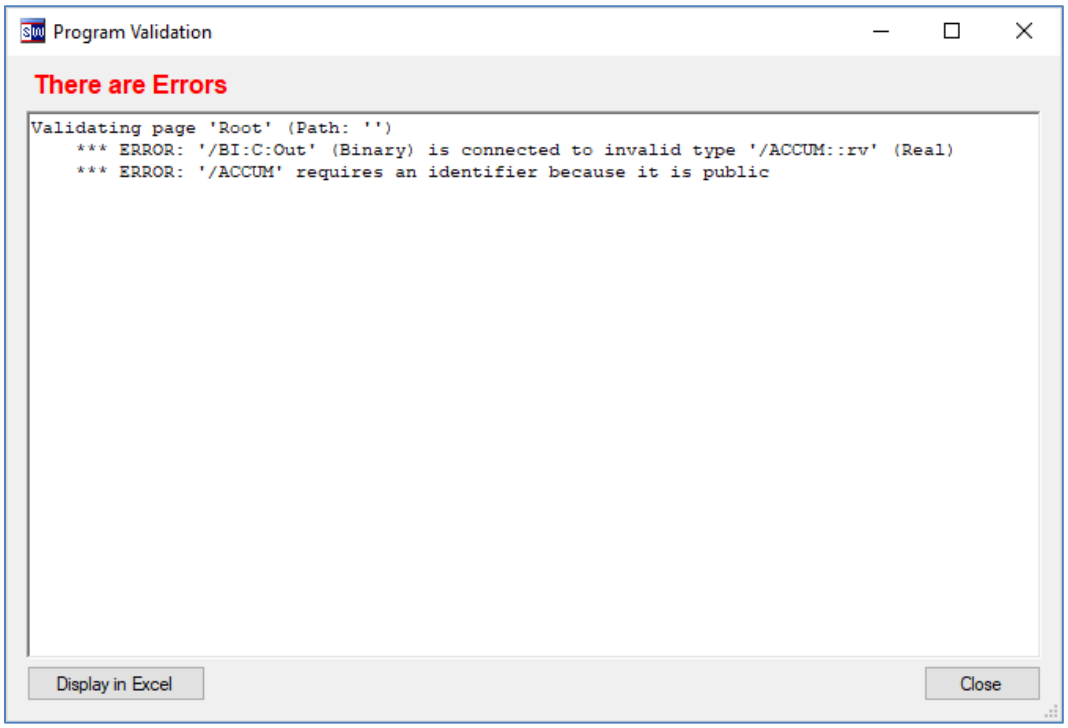
You can convert the Visio file back to an .AUT file at any time by clicking the Convert to .AUT button in the ribbon.

- The .AUT file will be saved in the same folder as the Visio file, overwriting the original.

Validation

The Studio Function Block editor validates your program before saving or uploading. There is also a Ribbon button for on-demand validation.





The validation checks to make sure your program follows certain rules:

Inputs

- All inputs must be connected. Programs that have a disconnected input will fail to compile in the original Function Block editor.

Data Types

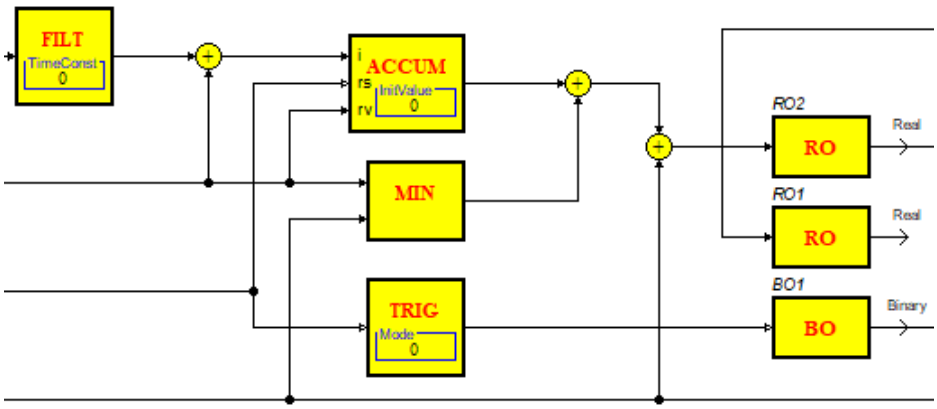
- Binary can connect to Binary.
- Integer can connect to Integer or Analog.
- Real can connect to Real or Analog.
- Analog can connect to Analog, Integer or Real.

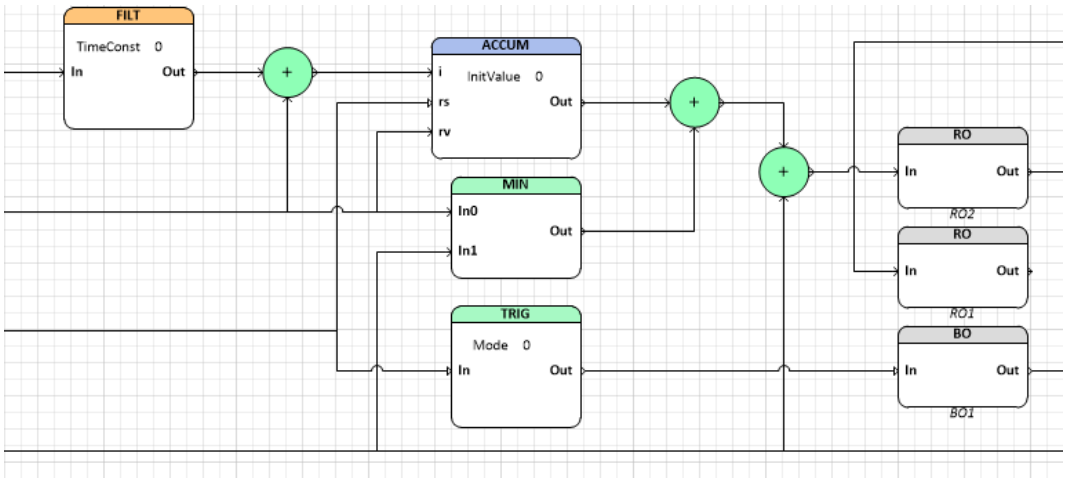
Identifiers

- Blocks marked as public must have an identifier.
- HFBs and HFB I/Os must have an identifier.
- Two or more blocks marked as public cannot share an identifier.
- Identifiers cannot exceed 20 characters, must start with a letter, and can contain letters, numbers and underscores.
- Prefixes cannot exceed 12 characters, and can contain letters, numbers and underscores.

Using Function Blocks

The Studio Function Block editor introduces a new set of Visio stencils that contain all of the standard Function Block blocks. These blocks contain the same information as their Function Block counterparts.





Some differences to note:

- Function Block uses connector dots to show where lines are connected and allow unconnected lines to cross. Visio creates wire jumps where unconnected lines cross, with intersecting lines indicating connection.
- If you move a connected block, the Visio routing will keep the connectors perpendicular (and not diagonal, as in Function Block). You can adjust individual line segments by clicking on them and dragging the center controls.
- In the standard Function Block editor, solid arrow heads indicate numeric values (integer or real), while empty arrow heads indicate binary values. Studio keeps this convention, but also uses a dot to differential double values from integer.

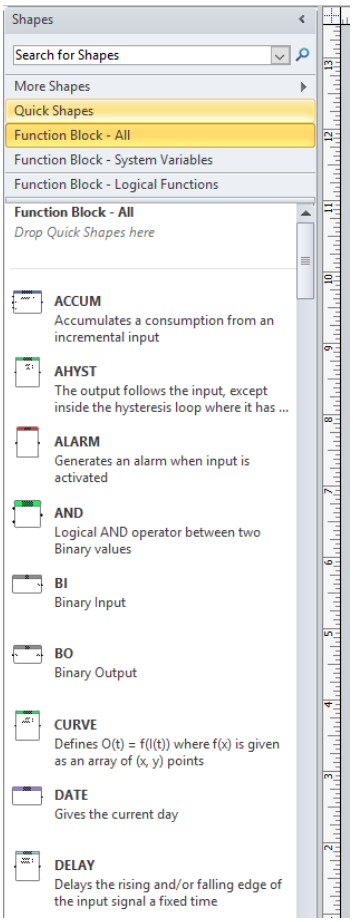
You can add blocks to a drawing in three different ways:

- You can drag a shape from any of the Function Block Visio stencils.
- You can select the block from the ADD menu in the Ribbon.
- You can right-click on the Visio drawing and use the ADD A FUNCTION BLOCK menu.

The Visio stencils are organized into a single stencil (*Function Block – All*) containing all the blocks alphabetically, plus another set of stencils grouping the blocks by category (e.g., *Function Block – Logical Functions*).

- The *All* stencil is generally shown automatically.
- The SHOW STENCIL menu in the Function Block Editor Ribbon can also be used to open up the subset stencils.

If you make the Shapes panel in Visio wide enough, and choose to View ICONS AND DETAILS (right-click on the Shape panel title bar), you can see a description of the blocks



Connecting Visio Function Blocks

You show the connections between the outputs of one block and the inputs of others by creating a Visio Connector line between them.

- On the *Function Block Editor* Ribbon, click the CONNECTOR icon to put Visio into Connection mode. This command is also available on the *Home* ribbon
- Drag the mouse cursor near the input or output of a block. A red square will highlight the connection point when you get near.
- Click the mouse button to start the Connection. Drag the line to the other input or output to finish the Connection.
- Click the POINTER TOOL icon in the ribbon to end Connection mode and return to the standard mode to move blocks as well as connection lines.

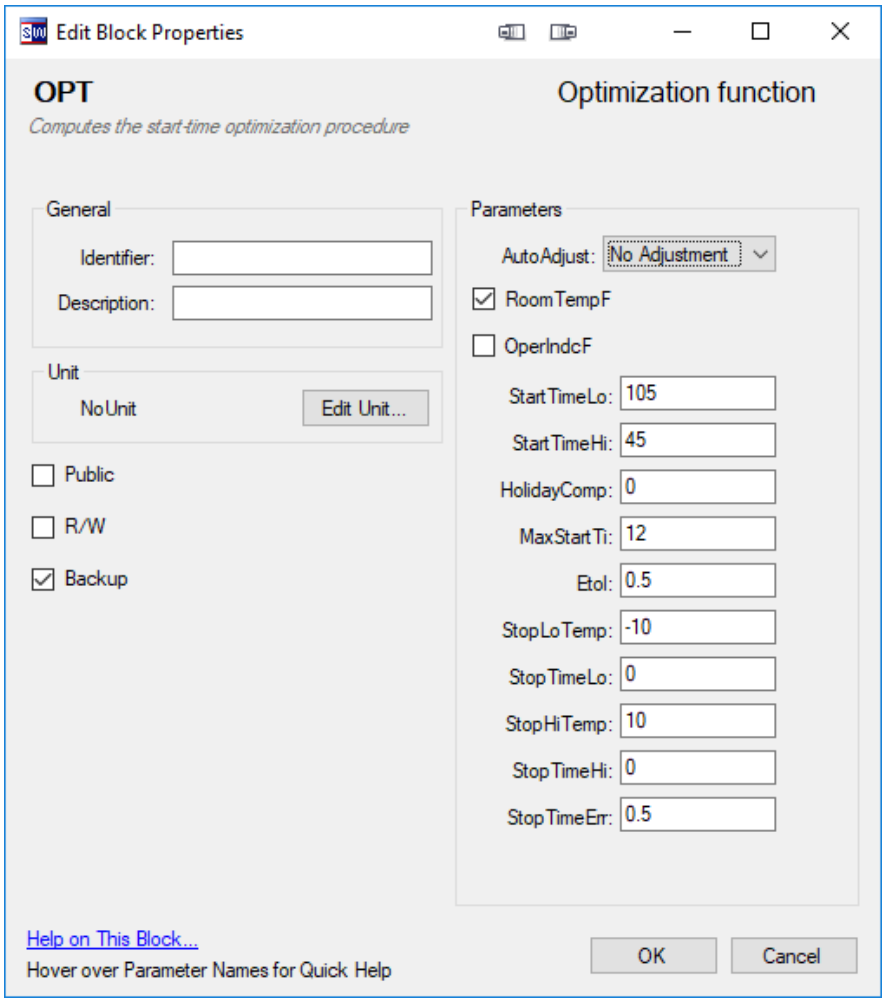
Editing Block Properties

To edit the properties of a block, you can

- Double-click the block.
- Right-click the block and select EDIT PROPERTIES from the menu.
- Select the block and click EDIT PROPERTIES in the ribbon.

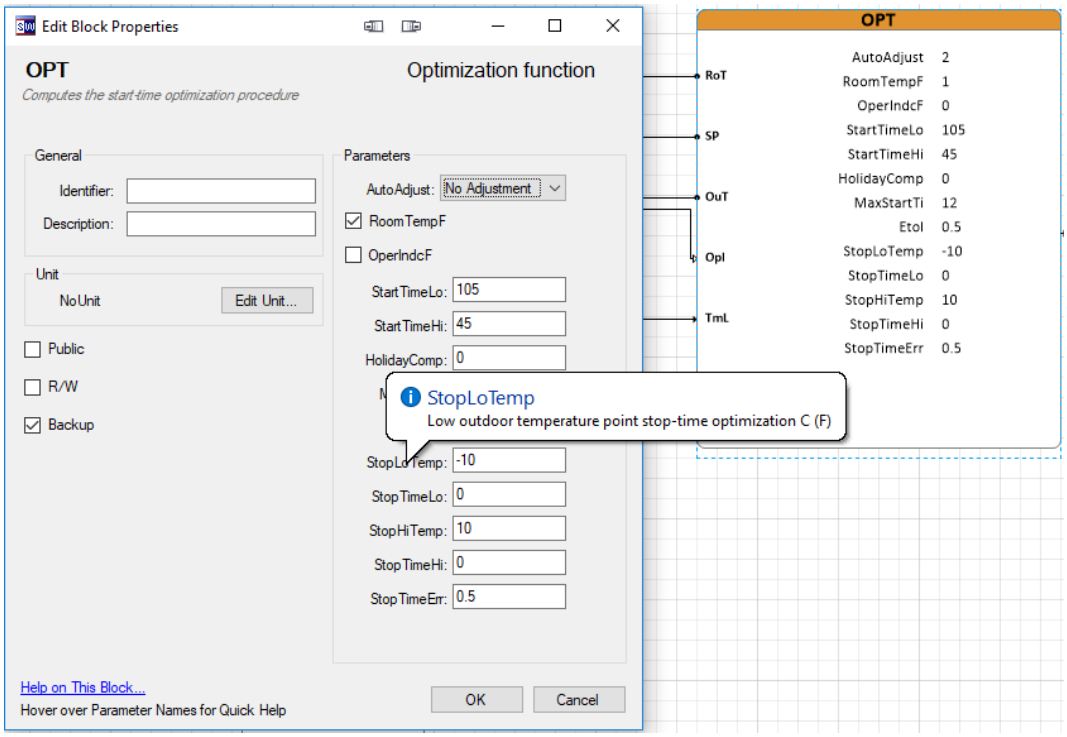
Some of the block's properties (PUBLIC, R/W, or MODE2 of an Operator) can be modified by right-clicking on the block and checking or unchecking menu options, where available.

The *Edit Block Properties* form contains all the information normally available in Function Block, plus some additional help information:



- There are additional descriptions of the block and its function at the top of the form.

- Click the HELP ON THIS BLOCK link to bring up a PDF of the block's page from the Function Block Reference manual.
- Hover over any parameter to see additional information about that parameter.



HFBs

An important part of any larger Function Block program is the use of Hierarchical Function Blocks (HFBs) to encapsulate portions of the program. They are used:

- To organize a larger program into a more structured one for easier editing, debugging and simulation.

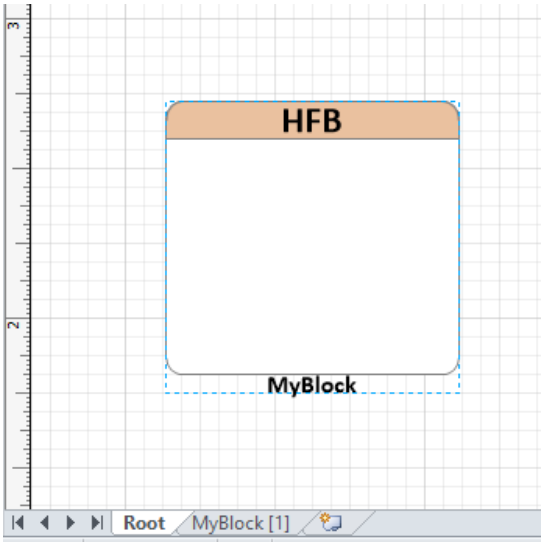
The Smartware Function Block Editor makes use of HFBs for additional features:

- To create new, reusable functions blocks that can be tested separately and stored in a library for easy searching and tracking version updates.
- To create a set of blocks that mimic the WorkPlace Tech programming blocks as much as possible, allowing for the conversion of WorkPlace Tech applications to Function Block programs.

In the standard Function Block Editor, you navigate in and out of the inside of the HFB through menu commands (EXPAND HFB and COMPRESS HFB).

In the Smartware Function Block Editor, Visio pages are used to navigate the HFBs.

- The first page is *Root*. This is the main part of the program.
- When you add a new HFB, you will be prompted for the name. A new Visio page will then be added for the contents of the HFB, using the name of the HFB, along with a number, as the page name.



- The number after the HFB name is used to differentiate between multiple HFBs in the file (on different pages) that have the same name.

You can navigate to the inside of an HFB by:

- Selecting the Visio page tab; or
- Right-clicking on the HFB block and selecting GO TO HFB PAGE; or
- Selecting the HFB and clicking GO INSIDE HFB in the Ribbon

When on an HFB page, you can navigate back to the HFB block by:

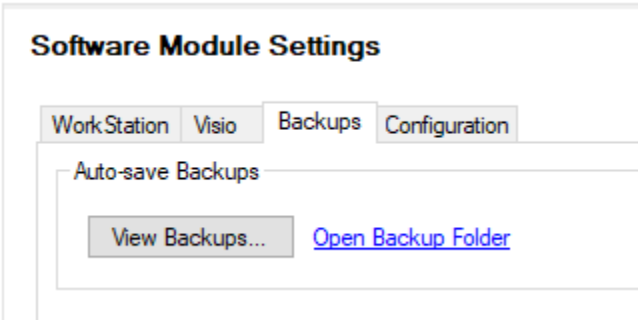
- Selecting the parent HFB's Visio page tab; or
- Right-clicking on the page and selecting GO TO PARENT HFB; OR
- Clicking GO TO PARENT HFB in the Ribbon

You should not manually create, delete or rename the Visio pages. All of these changes should occur automatically as you create, delete or rename the HFBs.

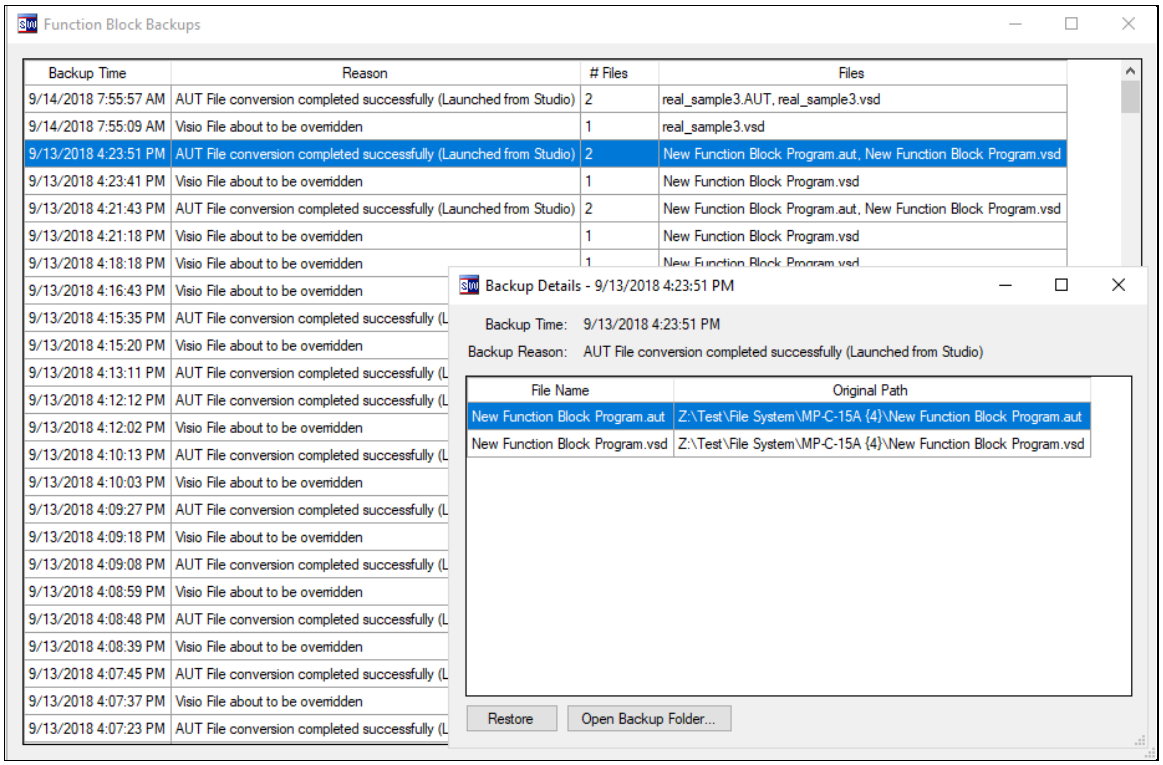
Auto-Saved Backups

There may be situations where a problem occurs when saving or loading the program file, especially when communicating with an EcoStruxure device or testing new features. To ensure that work isn't lost, Studio will make a backup of the .AUT and Visio drawing files at various points where a problem could occur.

To access the backups, go to TOOLS→OPTIONS and select the *Software Module* tab, and then select the *Backups* tab:



Click VIEW BACKUPS to see the details of the backed up files:



- Double-click an entry to view the details about the backup set.
- Click OPEN BACKUP FOLDER to open the specific folder in Windows Explorer.

- For files opened in Studio only, you can click RESTORE to put the original file back in its original location.

Copy and Paste

Most blocks can be copied and pasted without any issue. Since HFBs require additional logic in a separate page, copy and paste between instances of Visio are more complicated. In the Ribbon, there are options for special copy and paste commands, which will handle these cases.

4. The Custom Block Library

Smartware and your Company will be able to create, store, and distribute a library of HFB blocks that are easily used and reused within your Function Block programs.

Library Structure

Libraries in the editor do not contain any actual program, but contain information for configuring a versioned AUT file, which is the library source. When converting a program in the editor to AUT, the source file is loaded and tweaked according to various settings in the library block used. This can include:

- Identifier
- Description
- Constants can be exposed so they can be configured
    - Integer constants can be given ranges or converted to dropdown combo boxes

Each AUT file contains a comment in the root which contains additional information about the library, such as the version. This version is used by the editor to determine when library blocks are out of date, so they can be updated.

File System and Synching

Libraries are contained in folders that live inside *[APPDATA]/Function Block/Libraries*. Each library contains its own folder. Inside the folder are sub-folders that represent the categories for AUT files. Category folders can be nested. There are various files that are generated by the Function Block Editor during a syncing process.

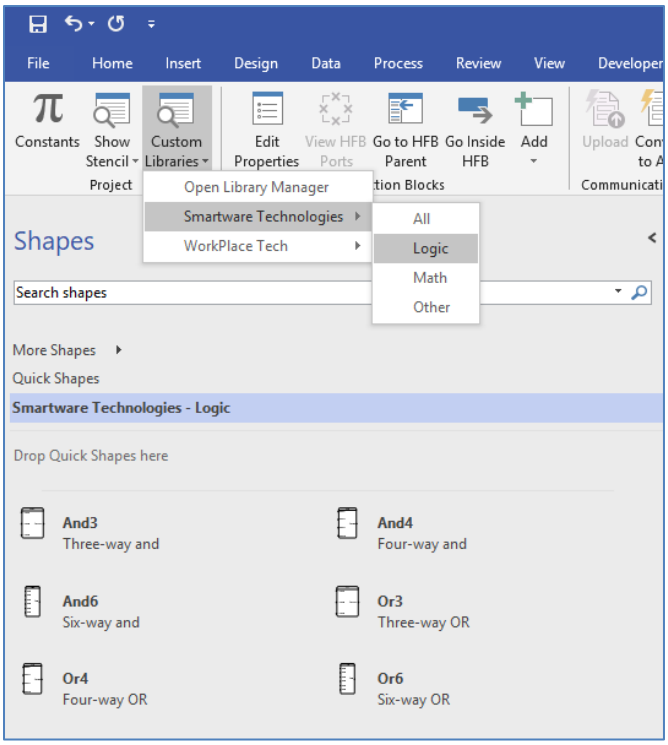
When a sync happens, each AUT file should have a **.meta.xlsx* file next to it, which contains information from scanning the respective file. These files, along with the AUT file itself, are what should be distributed.

In the root of the library folder is a master meta.xlsx file that contains all the information from individual **.meta.xlsx* files. The syncing process keeps this file up to date by looking at the timestamps on the individual **.meta.xlsx* files and updating itself when it finds a newer one. The root of the library folder also contains generated stencils.

Editor and Interface

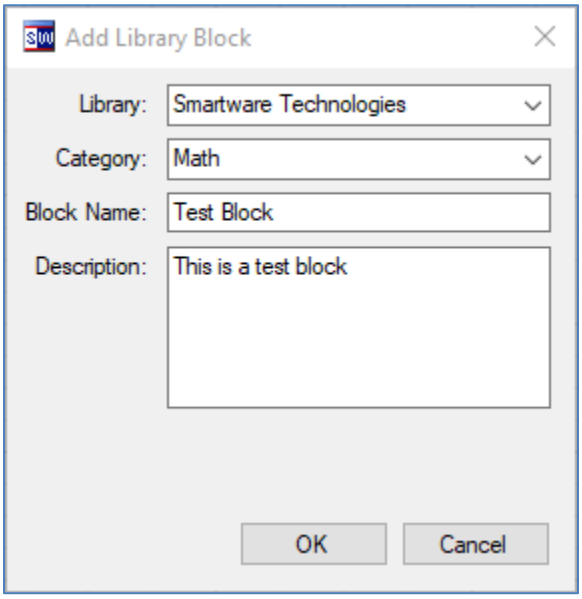
You can drop a library block shape by either right clicking the background of the drawing and selecting ADD A LIBRARY BLOCK SHAPE, or by dragging it from a docked library stencil. Docking can be done in a couple ways:

- When browsing the blocks from the ADD A LIBRARY BLOCK SHAPE menu select the DOCK STENCIL option; or
- In the ribbon, use the CUSTOM LIBRARIES menu to select a category to dock



Adding a New Library Block

You can convert any HFB into a library block by right clicking it and selecting CONVERT TO LIBRARY BLOCK. This will open a form asking you where you want your library to live.



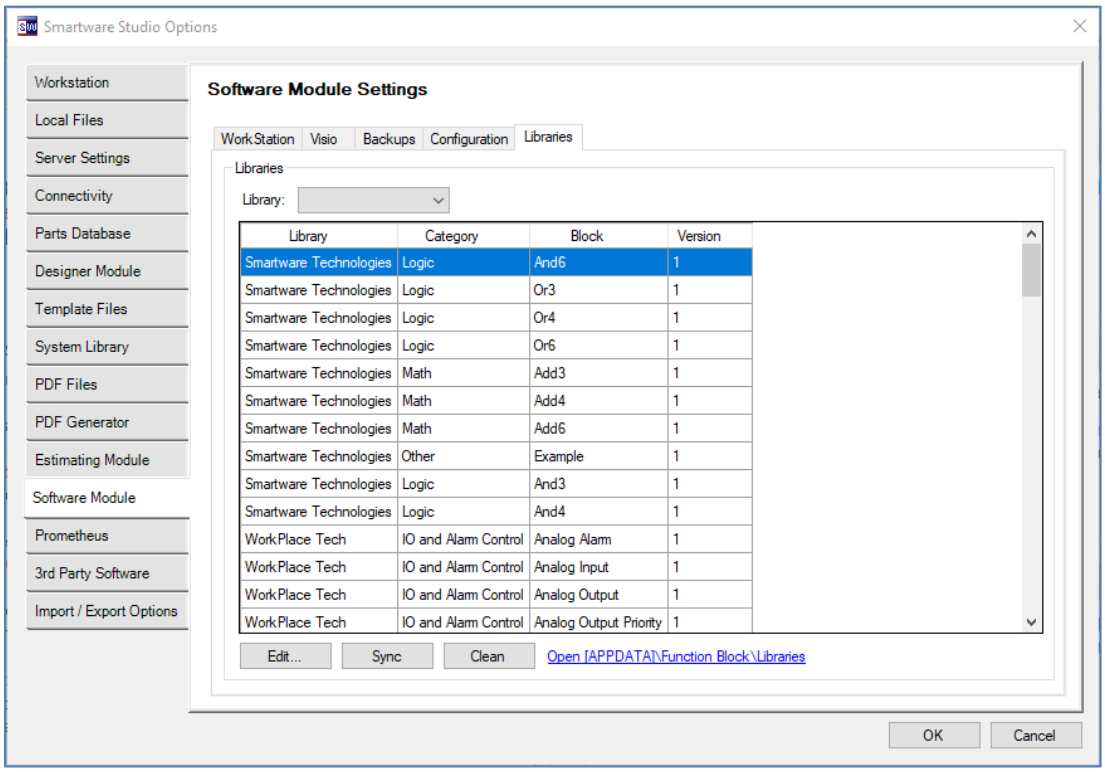
- You can choose to add it to an existing library or category, as well as create a new one.

Managing and Editing Libraries

You can open the Library Manager:

- In Studio: TOOLS → OPTIONS → SOFTWARE MODULE → LIBRARIES
- In the Function Block Editor Ribbon: CUSTOM LIBRARIES → OPEN LIBRARY MANAGER

The Library Manager shows all the blocks in the various custom libraries:



- In the Library Manager you can EDIT individual library blocks, SYNC ALL, or CLEAN ALL.
- Syncing your library collection will scan for newer *.aut* files whose changes aren't yet stored in the various **.meta.xlsx* files and update them accordingly. It will also handle updating stencils.
- Cleaning your library collection will clean out all **.meta.xlsx* files as well as stencils, which will make the editor no longer recognize the library until synced again. This should be used if the library gets into a bad state or is not syncing properly for any reason.

WorkPlace Tech and Smartware Library

We have included a small Smartware library, as well as a partial implementation of WorkPlace Tech blocks. Below is the current implementation status:

Name	Category	Status
Dual Delay	Timer and Sequence Control	Supported
Dual Minimum	Timer and Sequence Control	Supported
Event Indicator	Timer and Sequence Control	Supported
Interstage Delay (10)	Timer and Sequence Control	Supported

Interstage Delay (3)	Timer and Sequence Control	Supported
Interstage Delay (6)	Timer and Sequence Control	Supported
Minimum Off	Timer and Sequence Control	Supported
Minimum On	Timer and Sequence Control	Supported
Off Delay	Timer and Sequence Control	Supported
On Delay	Timer and Sequence Control	Supported
Sequence (10) [Analog]	Timer and Sequence Control	Supported
Sequence (10) [Linear]	Timer and Sequence Control	Supported
Sequence (10) [Vernier]	Timer and Sequence Control	Supported
Sequence (3) [Analog]	Timer and Sequence Control	Supported
Sequence (3) [Linear]	Timer and Sequence Control	Supported
Sequence (3) [Vernier]	Timer and Sequence Control	Supported
Sequence (6) [Analog]	Timer and Sequence Control	Supported
Sequence (6) [Linear]	Timer and Sequence Control	Supported
Sequence (6) [Vernier]	Timer and Sequence Control	Supported
Step Driver	Timer and Sequence Control	Planned

Name	Category	Status
Analog Alarm	IO and Alarm Control	Planned
Analog Input	IO and Alarm Control	Supported: RI
Analog Output	IO and Alarm Control	Supported: RO
Analog Output Priority	IO and Alarm Control	Planned
Binary Alarm	IO and Alarm Control	Planned
Binary Input	IO and Alarm Control	Supported: BI
Binary Output	IO and Alarm Control	Supported: BO
DUI Expander	IO and Alarm Control	Supported
Fan Speed	IO and Alarm Control	Supported
Floating Actuator	IO and Alarm Control	Planned
Floating Actuator Priority	IO and Alarm Control	Planned
Momentary Start / Stop	IO and Alarm Control	Supported
Pressure Transducer	IO and Alarm Control	Planned
PWM	IO and Alarm Control	Planned
PWM Priority	IO and Alarm Control	Planned
Sensor Input	IO and Alarm Control	Planned
VAV Actuator	IO and Alarm Control	Planned

Name	Category	Status
Abs Sub / Div	Logic and Math Control	Supported
Add / Add	Logic and Math Control	Supported
Add / Div	Logic and Math Control	Supported
AND / AND	Logic and Math Control	Supported
AND / OR	Logic and Math Control	Supported

Average	Logic and Math Control	Supported
Clocked SR	Logic and Math Control	Supported
Compare	Logic and Math Control	Supported
Compare2	Logic and Math Control	Supported
Count Down	Logic and Math Control	Supported
Count Up	Logic and Math Control	Supported
Curve Fit	Logic and Math Control	Planned
Enthalpy	Logic and Math Control	Planned
EXOR	Logic and Math Control	Supported
Filter	Logic and Math Control	Supported
Latch	Logic and Math Control	Supported
MA Volume	Logic and Math Control	Supported
Mul / Add	Logic and Math Control	Supported
Mul / Div	Logic and Math Control	Supported
OR / AND	Logic and Math Control	Supported
OR / OR	Logic and Math Control	Supported
SqRt Mul / Add	Logic and Math Control	Supported
SR Flip-Flop	Logic and Math Control	Supported
Sub / Add	Logic and Math Control	Supported
Sub / Div	Logic and Math Control	Supported
Sub / Mul	Logic and Math Control	Supported
Sub / Sub	Logic and Math Control	Supported

Name	Category	Status
Binary Encoder	Loop and Process Control	Supported
Control Override	Loop and Process Control	Supported
COV Priority	Loop and Process Control	Needs Discussion
Demux Select	Loop and Process Control	Supported
High Select	Loop and Process Control	Supported
Interlock (Automatic)	Loop and Process Control	Supported
Interlock (Manual)	Loop and Process Control	Supported
Limit	Loop and Process Control	Supported
Limit Thermostat	Loop and Process Control	Supported
Loop Sequenced	Loop and Process Control	Planned
Loop Single	Loop and Process Control	Supported
Low Select	Loop and Process Control	Supported
Priority Input (2)	Loop and Process Control	Needs Discussion
Priority Input (4)	Loop and Process Control	Needs Discussion
Priority Value Select	Loop and Process Control	Needs Discussion
Ramp	Loop and Process Control	Supported
Reset	Loop and Process Control	Supported
Select	Loop and Process Control	Supported

Setpoint Control	Loop and Process Control	Planned
Thermostat	Loop and Process Control	Supported
Thermostat 2	Loop and Process Control	Supported

Name	Category	Status
Calendar	Schedule Control	Needs Discussion
OSS	Schedule Control	Needs Discussion
Schedule 7-Day	Schedule Control	Needs Discussion

5. Converting WorkPlace Tech Applications to Function Block

Note: *This feature is under development. You can enable it in its current state in the TOOLS→OPTIONS→SOFTWARE MODULE→CONFIGURATION tab.*

Studio can read a WorkPlace Tech Visio application file and translate it into a Studio Function Block Visio file.

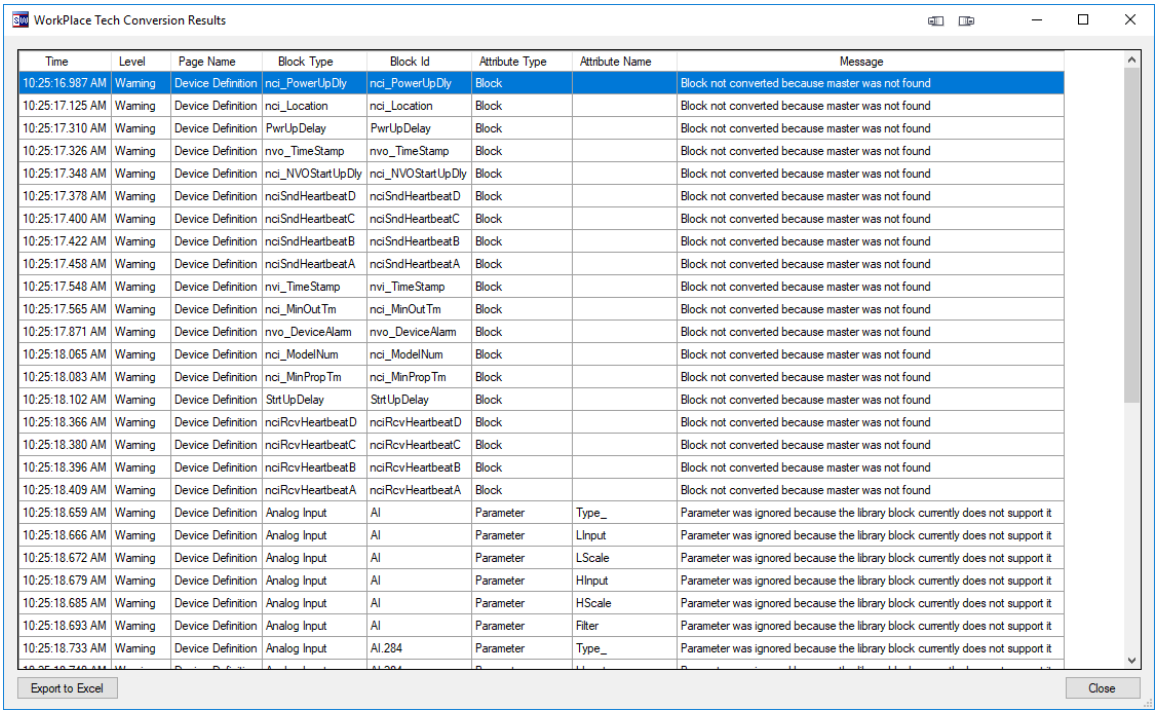
- Each WorkPlace Tech block will be replaced with a corresponding block from our library that contains the same inputs and outputs.
    - These versions mimic the WorkPlace Tech features, and do not rely on less-featured Function Block versions.
    - For example, the Function Block HYST block is similar to the WorkPlace Tech TSTAT block, but the TSTAT contains additional inputs for *Setpoint* (Setpt) and *Input Differential* (InDiff).
- The conversion tool uses the WorkPlace Tech Function Block Library, which is not fully implemented yet. Conversions can continue, but errors will be generated when using blocks that aren't fully implemented yet. These blocks will be replaced by template HFBs (without code) instead of library blocks so that the user can implement them if they wish.
- Function Block does not allow unused inputs, but WorkPlace Tech does. In some cases, the conversion tool can provide sensible defaults, but many times it cannot.
    - For example, on an Analog Input, the first input goes to a physical input and the second goes to a value that produces an offset to apply to the input before use. If no offset is provided, then we include a constant that contains zero.

Conversion and Logs

To convert a WorkPlace Tech program to a Function Block program, right click the program from within Studio and select CONVERT TO FUNCTION BLOCK. After the conversion is completed, a log will provide details on what happened and what still needs to be addressed.

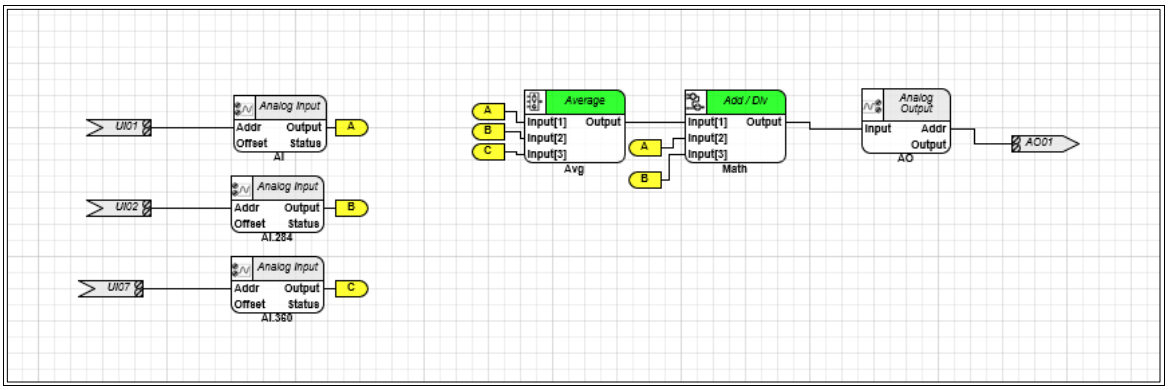
- Info level logs give general information about the conversion.
- Warning level logs tell the user that something may require further attention, but it may not cause any issues.
    - For example, a block had an identifier that is not allowed in Function Block and was renamed.

- Error level logs tell the user that the converted program will not work unless these issues are taken care of.
    - For example, a block that is not yet implemented in the WorkPlace Tech library was used, so the resulting program will not yet work.

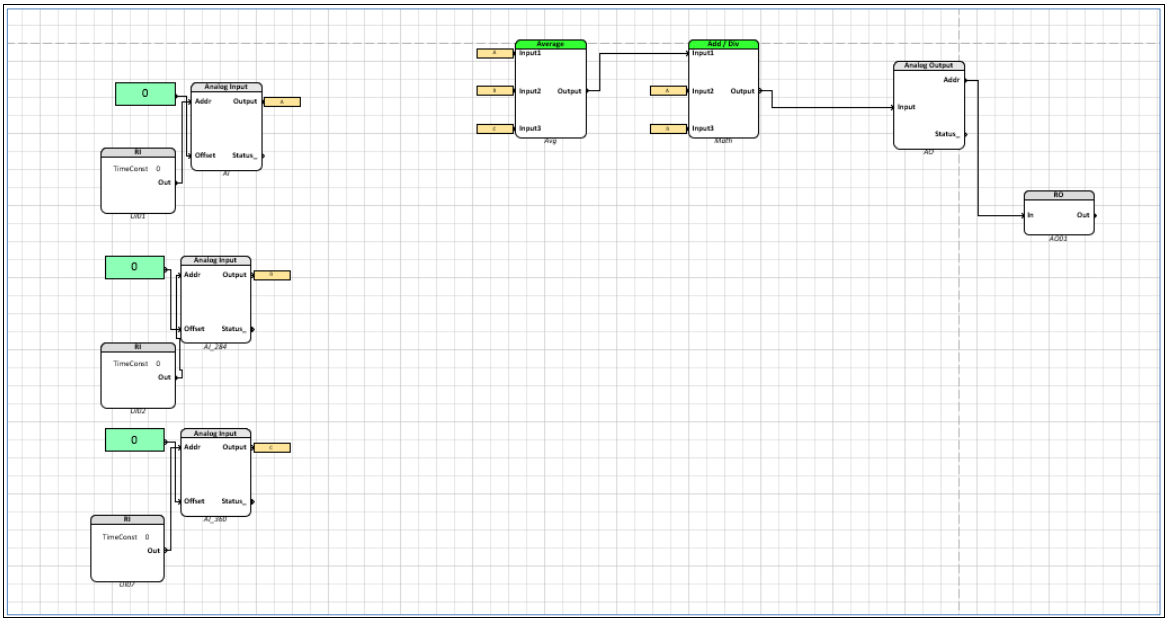


Time	Level	Page Name	Block Type	Block Id	Attribute Type	Attribute Name	Message
10:25:16.987 AM	Warning	Device Definition	nci_PowerUpDly	nci_PowerUpDly	Block		Block not converted because master was not found
10:25:17.125 AM	Warning	Device Definition	nci_Location	nci_Location	Block		Block not converted because master was not found
10:25:17.310 AM	Warning	Device Definition	PwrUpDelay	PwrUpDelay	Block		Block not converted because master was not found
10:25:17.326 AM	Warning	Device Definition	nvo_TimeStamp	nvo_TimeStamp	Block		Block not converted because master was not found
10:25:17.348 AM	Warning	Device Definition	nci_NVOStartUpDly	nci_NVOStartUpDly	Block		Block not converted because master was not found
10:25:17.378 AM	Warning	Device Definition	nciSndHeartbeatD	nciSndHeartbeatD	Block		Block not converted because master was not found
10:25:17.400 AM	Warning	Device Definition	nciSndHeartbeatC	nciSndHeartbeatC	Block		Block not converted because master was not found
10:25:17.422 AM	Warning	Device Definition	nciSndHeartbeatB	nciSndHeartbeatB	Block		Block not converted because master was not found
10:25:17.458 AM	Warning	Device Definition	nciSndHeartbeatA	nciSndHeartbeatA	Block		Block not converted because master was not found
10:25:17.548 AM	Warning	Device Definition	nvi_TimeStamp	nvi_TimeStamp	Block		Block not converted because master was not found
10:25:17.565 AM	Warning	Device Definition	nci_MinOutTm	nci_MinOutTm	Block		Block not converted because master was not found
10:25:17.871 AM	Warning	Device Definition	nvo_DeviceAlarm	nvo_DeviceAlarm	Block		Block not converted because master was not found
10:25:18.065 AM	Warning	Device Definition	nci_ModelNum	nci_ModelNum	Block		Block not converted because master was not found
10:25:18.083 AM	Warning	Device Definition	nci_MinPropTm	nci_MinPropTm	Block		Block not converted because master was not found
10:25:18.102 AM	Warning	Device Definition	StrtUpDelay	StrtUpDelay	Block		Block not converted because master was not found
10:25:18.366 AM	Warning	Device Definition	nciRcvHeartbeatD	nciRcvHeartbeatD	Block		Block not converted because master was not found
10:25:18.380 AM	Warning	Device Definition	nciRcvHeartbeatC	nciRcvHeartbeatC	Block		Block not converted because master was not found
10:25:18.396 AM	Warning	Device Definition	nciRcvHeartbeatB	nciRcvHeartbeatB	Block		Block not converted because master was not found
10:25:18.409 AM	Warning	Device Definition	nciRcvHeartbeatA	nciRcvHeartbeatA	Block		Block not converted because master was not found
10:25:18.659 AM	Warning	Device Definition	Analog Input	AI	Parameter	Type_	Parameter was ignored because the library block currently does not support it
10:25:18.666 AM	Warning	Device Definition	Analog Input	AI	Parameter	LInput	Parameter was ignored because the library block currently does not support it
10:25:18.672 AM	Warning	Device Definition	Analog Input	AI	Parameter	LScale	Parameter was ignored because the library block currently does not support it
10:25:18.679 AM	Warning	Device Definition	Analog Input	AI	Parameter	HInput	Parameter was ignored because the library block currently does not support it
10:25:18.685 AM	Warning	Device Definition	Analog Input	AI	Parameter	HScale	Parameter was ignored because the library block currently does not support it
10:25:18.693 AM	Warning	Device Definition	Analog Input	AI	Parameter	Filter	Parameter was ignored because the library block currently does not support it
10:25:18.733 AM	Warning	Device Definition	Analog Input	AI.284	Parameter	Type_	Parameter was ignored because the library block currently does not support it

Here is the original WorkPlace Tech application:



And here is the generated Function Block program:



Due to the differences in sizes and of the WorkPlace Tech shapes and their Function Block equivalents, some of the layout will need adjustment.

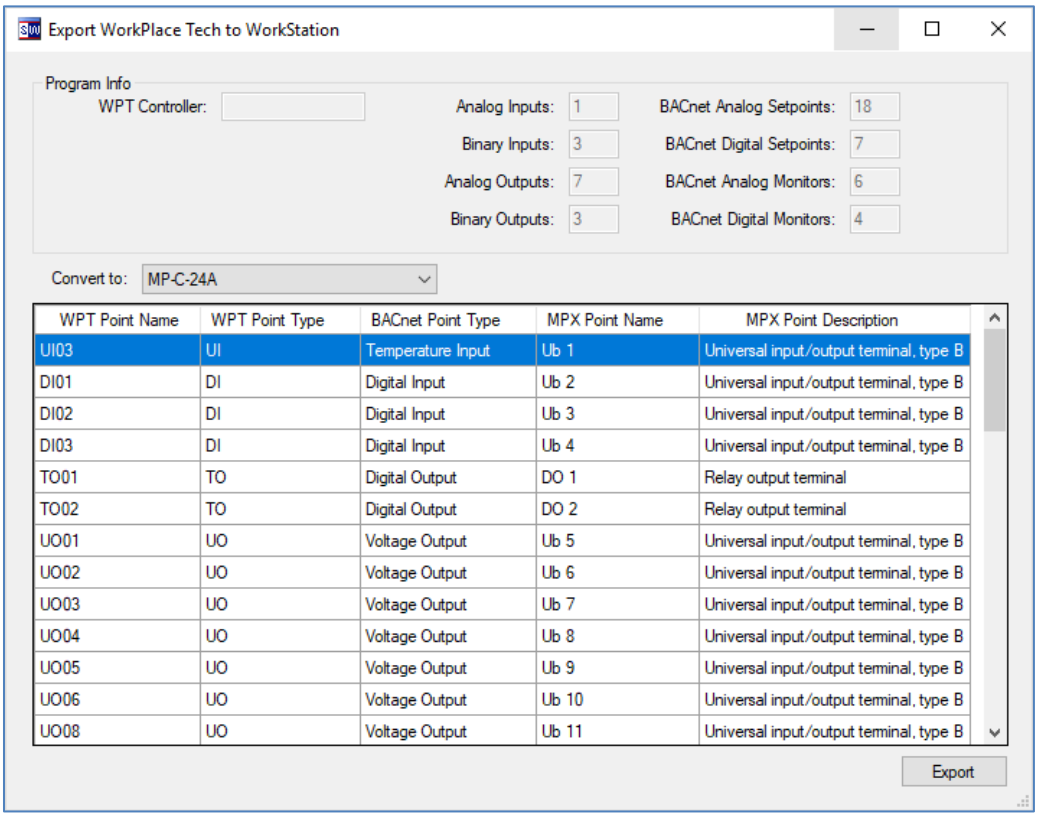
- We expect to improve this aspect in later versions.

Inputs, Outputs, BACnet and WorkStation

In WorkPlace Tech, there are physical input and output points, and blocks to work with them. These do not have a translation to Function Block, because Function Block has no concept of physical points. For example, an Analog Input may specify the scale for reading a voltage it receives from a physical point. Even though Function Block doesn't have this concept, WorkStation does.

To create an equivalent Function Block program that deals with physical inputs and outputs, supplementary WorkStation inputs and outputs are required. These inputs and outputs can contain information such as scaling voltage or current and can be bound to a Function Block input. Because this information must be split between Function Block and WorkStation, we have provided a utility for generating an MP-series controller in WorkStation based on the output of a converted WorkPlace Tech program.

After the logic is converted to Visio, you will see the Export WorkPlace Tech to Workstation options:



- The number of inputs and outputs are counted, and the type of input or output block is analyzed. Information like scaling, offset and input sensor type is recorded.
- Based on the information gathered, the conversion tool decides on WorkStation point types (BACnet Point Type) for each input and output in the program and configures it. For example, an input sensor type of *Thermistor (10k)* will create a WorkStation Temperature Input, and the COV Increment and scaling information will be copied from the WorkPlace Tech Analog Input to the WorkStation Temperature Input.
    - In the converted program, these become regular input/output objects, such as RI, BI, RO, BO.
    - Special-purpose I/O blocks are handled on a per-case basis. Some of these blocks will generate an I/O block as well as a library block handling the custom logic, such as a *Momentary Start / Stop*.
- Based on the types of each physical point, the conversion tool comes up with a list of possible MP-series controllers that are compatible with the program and will allocate the WorkStation points.

- BACnet points are not allocated, but the points exist.
    - In the converted program, these become regular input/output objects, such as RI, BI, RO and BO.
- Exporting the current configuration generates an XML file that can be imported into WorkStation. This file contains the application folder. It does not currently contain the program and will require manual upload and binding afterwards.

Conversion Considerations

We are still in the process of creating all the Function Block versions of the WorkPlace Tech blocks, but there will be certain features of WPT that won't translate to Function Block. We will work to handle these in a consistent and logical manner.

We look forward to any insights you have on how these should be handled.

The following sections note the known issues.

For a full, comprehensive list of the implementation status and technical details, see Appendix A.

N/A Values

We currently do not have a good way to deal with N/A values. If you have any suggestions or have a way of doing this currently, please let us know.

Priority Values

We currently do not have a good way to deal with Priority values. If you have any suggestions or have a way of doing this currently, please let us know.

No Physical Points

Function Block doesn't have the concept of a physical point, which leads to a couple major differences in converted programs:

- There are no Address inputs or outputs, since that would imply that these connect to a physical point. Instead, these inputs and outputs either connect directly to an RI/BI/RO/BO block, or the block itself is converted to an RI/BI/RO/BO block. The *Analog Input* block is one case where the entire block is converted to an RI block.
- Blocks that contain both an Address output as well as a data output are redundant, since Function Block can only output data. Blocks like these are redesigned to only have a single data output, and all connected outputs are rerouted. For example, a *Momentary Start / Stop* has both an Address output as well as a data output. The output of the *Momentary Start / Stop* library block connects to both a Function Block output and what the data output was connected to, if any.

Layout

Function Block blocks have a different layout compared to WorkPlace Tech blocks, making 1-to-1 conversion layouts very difficult without being very spaced out. Right now, we are trying out a group layout strategy, which tries to split a WorkPlace Tech program into visual groups based on tags. Blocks that are connected by a physical connector are considered part of the same group. These groups are then laid out in to columns.

6. Advanced Features

Remote Tags

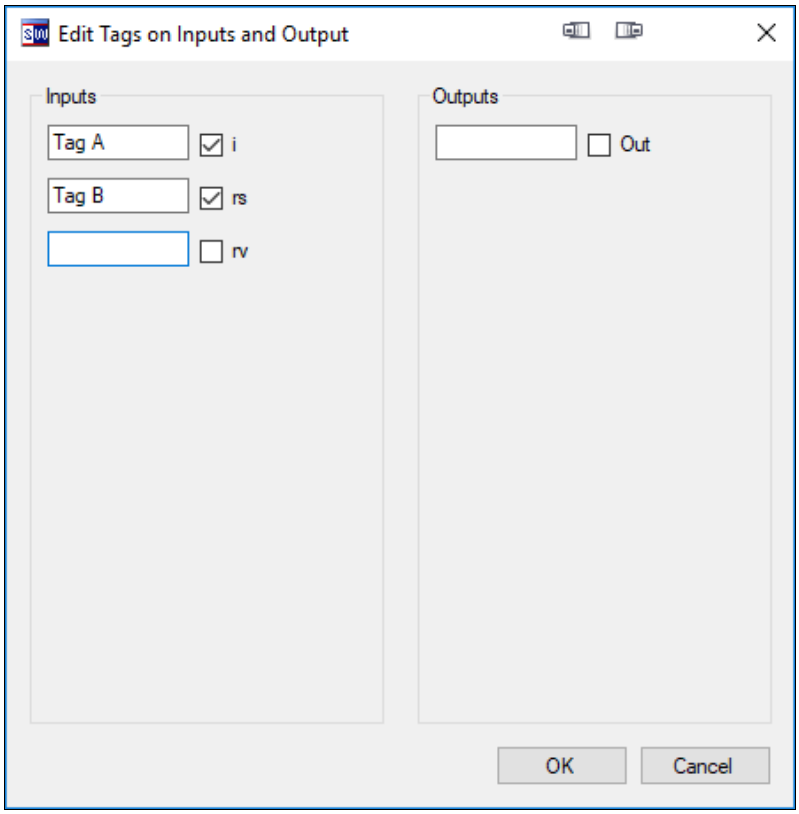
WorkPlace Tech has a feature that allows you to place a tag on the output of one block and another tag on the input of another block. This creates an implicit connection without having to see the connection line go across the page in cases where the two blocks are not near one another.

The Studio Function Block Editor implements this concept by showing the tags and hiding the connecting line when editing but replacing the connection line when saving back to Function Block (where it's required).

- When using tags, blocks can still be moved around freely, and Visio takes over with the automatic routing of the invisible connectors as usual. The programmer may want to adjust this routing after the application is finished, as it will then be viewed more frequently in the Graphics Viewer in WorkStation as Function Blocks.

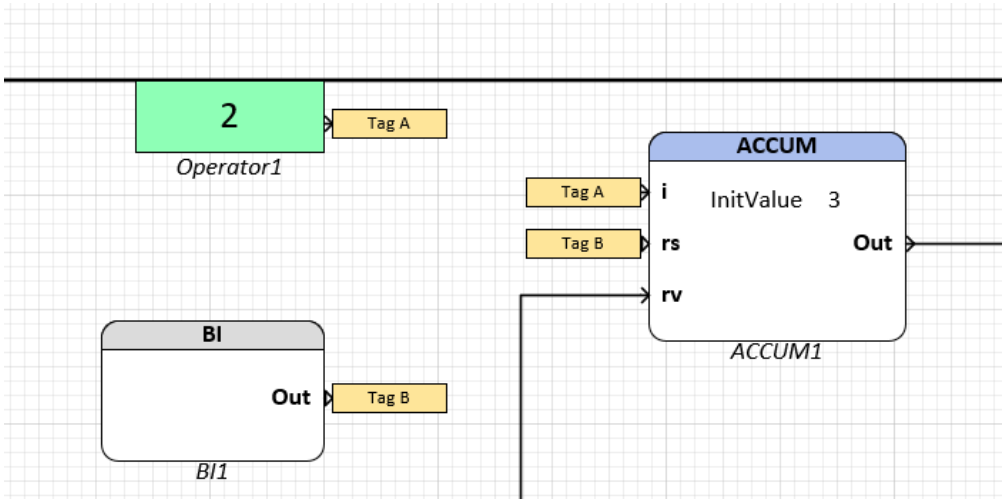
To replace a connection with a pair of tags

- Right-click on a block and select EDIT TAGS:



- For the inputs or outputs you want to show as tags, check the box and specify the text to show in the tag.
- When setting a tag on one port, the connection is automatically followed and tags will appear on all other connected ports.

The tags will appear on the inputs and outputs.



- You can uncheck the Show Tag box at either end of a connection to hide the tag and reshow the connecting line.
- There will eventually be a button in the ribbon to temporarily hide all tags. This will help verify the layout for the function block version without manually hiding individual tags.

Using a Single HFB Multiple Times in an Application

For your custom HFB blocks in an application, you will be able to create them once in your application and then reference them multiple times.

- If you need to modify the details of the HFB, you only will need to do it in one place.
- When the program is converted back to an AUT file, the details of the HFB will be repeated once for each instance (as is required).
- If you decide you want to customize an instance of the HFB, you will be able to easily duplicate and separate its detail.

VSDM (Flex) Conversion vs. VSD (Interop) Conversion

There are two types of conversions available in the editor. The original VSD conversion works with Visio 2010 and later, and is very slow. We have added a new converter that works with VSDM files, which is significantly faster.

The new converter is compatible with Visio 2013 and up. For AUT to Visio conversion, it may be possible for Visio 2010 users to take advantage of this if they have Visio Service Pack 2 (SP2) installed. Converting from Visio to AUT still requires Visio 2013 or later. The VSDM converter is currently under development. Overall, the VSDM converter is much faster than VSD, and should always be preferred. If you have the appropriate version of Visio installed and the VSDM conversion is enabled, then the VSDM converter will automatically be used. It can be disabled in the Studio options (Tools→Options→Software Module→Configuration).

There are three places where Interop or Flex conversions can be used. Here is the current support table for it:

Step	**VSD (Interop)**	**VSDM (Flex)**
AUT to Visio Conversion	Visio 2010+	Visio 2010 SP2+
Visio to AUT Conversion	Visio 2010+	Visio 2013+
Document Scanning	Visio 2010+	Visio 2013+

The Document Scanning step happens when a document is opened and is used to check for updates. The longer this step takes, the longer it takes for a document to open. When using the older VSD scanner, the startup time for a document after conversion can take up to 10 seconds for a very large file. If this becomes a problem, the auto-update feature can be disabled in the Studio options. The VSDM scanner is much faster.

7. Testing Considerations

If a problem occurs editing a file from EBO or Studio:

1. Check the Auto-Save Backups to locate any potentially lost work.
    a. Refer to Chapter 3 for details

2. Studio might leave a dead process around that could interfere with further edits. You may need to kill them from the Windows Task Manager.

    a. Depending on your version of Windows, they will appear on the Details tab, or possibly on the Applications tab as a child process of EBO Workstation

    b. If Visio is frozen, kill the *Visio.exe* process

    c. If opening from EBO, you might need to kill the process involved in converting the Visio file:
        i. *Smartware Studio.exe* (or *Studio360.exe*), with less than 50 MB of memory in use. More than 200 MB indicates a running instance of Studio.
        ii. *tam32s.exe*

The validation of the Visio version of the program is not fully complete. It may let you save programs that won't compile in EBO. When you attempt to upload these files to EBO, you will get the same error message that Function Block normally generates, which is unfortunately not very detailed.

- An example would be a missing or invalid input on a block.
- We will be adding our own validation to make these errors easier to resolve than is possible in standard Function Block.
- If you find specific validation cases that need to be caught, please report them to us so we can include them.

There are many experimental features that may potentially have problems with them. If any problem arises which prevents any kind of work from being done in the editor while that feature is active, it can be disabled in the Studio options.

8. Appendix A

This section goes over the details on how WorkPlace Tech programs are converted to Function Block, as well as the known differences between the two programs. This section is a work in progress and will be updated regularly.

General Differences

This section details general differences between WorkPlace Tech programs and generated Function Block programs.

NA Values

Function Block does not have the concept of NA values. Blocks that depend on NA values to perform certain functions are either implemented as multiple separate library blocks, or cannot be converted at all. In certain cases, NA values can be substituted for a constant where they would behave the same way in WorkPlace Tech.

Priority

Because Function Block doesn't have the concept of NA values, priorities are also not supported.

Layout

Function Block implementations of WorkPlace Tech blocks use HFBs. Because of this, the port placement and block size is very different. Attempts at laying out the resulting program are still being made. The current implementation uses a group layout strategy:

1. Split the WorkPlace Tech blocks into groups. Groups are defined by tag placement, as well as general connectivity. If two blocks are physically connected in WorkPlace Tech with a line, then they are a part of the same group.
2. Lay out each group.
    a. A block is selected from the group as a starting point, and becomes the first block in the current column.
    b. All blocks that connect to its inputs are laid out in a column to the left.
    c. All blocks that connect to its outputs are laid out in a column to the right.
    d. The left and right blocks are processed the same way.
3. Lay out the groups.
    a. The bounds of each group is calculated, and are laid out from left to right.
    b. There is a maximum width. Once exceeded, the next group will create a new row.

A previous attempt involved a naïve 1:1 coordinate scaling strategy. All coordinates were multiplied by a constant to attempt to make room for all blocks. This resulted in very spaced out programs, but had the advantage that the program structure was the same.

Inputs and Outputs

This section details how various inputs and outputs are converted. This section deals with blocks that have an address input or address output that normally tie to a hardware point. These points generate both Function Block inputs and outputs, as well as WorkStation points. For each physical point, a WorkStation virtual input/output must be created, and a compatible physical point type.

Analog Alarm

This block is currently not supported.

Analog Input

This is converted to a Real Input in Function Block. Various properties in the Analog Input block are used to generate a WorkStation Input.

Parameters

COV Increment: This is copied if the generated WorkStation Input supports this field.
Scaling Parameters: This is copied to the WorkStation Engineering / Electrical scaling parameters if the generated WorkStation Input supports these fields.
Offset: If the offset is not used or the offset is a constant, then the offset value (or zero, if not connected) is copied to the Offset field in the WorkStation Input if supported.
Sensor Type: The Sensor Type determines the type of WorkStation point.

WorkStation Point Selection

The Platinum Sensor Type is currently not dealt with.

WorkPlace Tech	WorkStation Point
Thermistor (10k)	Temperature Input
Balco	Temperature Input
Platinum	???
Milliamps	Current Input
Volts	Voltage Input
Resistance 1k	Resistive Input
Resistance 10k	Resistive Input

Physical Point Type Preferences

1. Universal Input Terminal
2. Universal IO Terminal Type B
3. Universal IO Terminal Type C

Analog Output

This is converted to a Real Output in Function Block. The output of the WorkPlace Tech block is tied to the output of the Real Output block. Attached Command Priorities are supported. For more information, see the section on Command Priority.

Parameters

COV Increment: This is copied if the generated WorkStation Input supports this field.
Scaling Parameters: This is copied to the WorkStation Engineering / Electrical scaling parameters if the generated WorkStation Input supports these fields.

WorkStation Point Selection

This currently generates only a WorkStation Voltage Output.

Physical Point Type Preferences

1. Analog Output Terminal
2. Universal IO Terminal Type B
3. Universal IO Terminal Type C

Analog Output Priority

This block is currently not supported.

Binary Alarm

This block is currently not supported.

Binary Input

This is converted to a Binary Input in Function Block.

Parameters

Type: The polarity is copied if the generated WorkStation Input supports this field. The Pulse type is currently not supported.

WorkStation Point Selection

This currently generates only a WorkStation Digital Input.

Physical Point Type Preferences

1. Digital Input Terminal
2. Universal Input Terminal
3. Universal IO Terminal Type B
4. Universal IO Terminal Type C

Binary Output

This is converted to a Binary Output in Function Block. The Binary Action parameter does not currently affect the output of the Binary Output block. The output of the

WorkPlace Tech block is tied to the output of the Real Output block. Attached Command Priorities are supported. For more information, see the section on Command Priority.

Parameters

Binary Action: This parameter is copied to the WorkStation Output.

WorkStation Point Selection

This currently generates only a WorkStation Digital Output.

Physical Point Type Preferences

The type of physical point required depends on the type of physical point used.

WorkPlace Tech	Required Physical Point
TO	Triac Output / Relay / High Power Relay
DO	Relay / High Power Relay

Command Priority

Command Priority attachments are supported. When used, instead of generating a single output (RO/BO), as many outputs as used are created. Because the output of the Command Priority is dependent on which value is used, which in turn is dependent on which priority was the highest, the output of this is entirely ignored.
Since Function Block programs don't have a concept of priority, this should only be used temporarily since every value fed into it will always be non-NA, resulting in a single value always being used, regardless of any priority that was attached to it in WorkPlace Tech.

BACnet Values

This section details how BACnet points are converted. These points generate both Function Block inputs and outputs, as well as WorkStation values.

Analog COV Client

This is converted to a Real Input in Function Block. In WorkStation, an Analog Consumer Value is generated. The COV Increment parameter is copied.

Analog Monitor

This is converted to a Real Output in Function Block. In WorkStation, an Analog Value is generated. The COV Increment parameter is copied.

Analog Setpoint, Analog SP Priority

This is converted to a Real Input in Function Block. In WorkStation, an Analog Value is generated. The COV Increment parameter is copied.

Binary COV Client

This is converted to a Binary Input in Function Block. In WorkStation, a Digital Consumer Value is generated. The COV Increment parameter is copied.

Binary Monitor

Binary Setpoint, Binary SP Priority This is converted to a Binary Input in Function Block. In WorkStation, a Digital Value is generated. The COV Increment parameter is copied.

General Blocks

This section details general WorkPlace Tech block implementations. Most of the implementations try to get as close as possible to an acceptable replacement for the given WorkPlace Tech block in terms of usage and behavior. Since much of the behavior is the same, this section will highlight differences and technical details for the implementations.

Clocked SR

The Clocked SR block behaves the same way as it does in WorkPlace Tech.

Compare

The Compare block behaves the same way as it does in WorkPlace Tech.

Compare2

The Compare2 block behaves the same way as it does in WorkPlace Tech.

Control Override

Count Down

Count Up

COV Priority

Curve Fit

Demux Select

Dual Delay

Dual Minimum

Event Indicator

Filter

The Filter block applies an exponential low-pass filter that follows the same rule as the WorkPlace Tech block: Output = Previous Output + [Filter Constant * (Input – Previous Output)].
Due to the speed differences between the WorkPlace Tech Filter block and Function Block program cycles, the Function Block implementation performs 4 low-pass filter passes per program cycle. To help the filter algorithm converge to the target value as the values get smaller, the effective filter constant is overridden.

The percent difference is used to determine the filter constant:

%Diff	Effective Filter
>= 0.4	No change
0.2 <= %D < 0.4	0.50
0.4 <= %D < 0.1	0.75
<= 0.1	1

High Select

Interlock

The Interlock block has two implementations, depending on if the Reset input is used. The difference in reset logic was significant enough that it was separated into two blocks.

> **Automatic Reset**: If the WorkPlace Tech Interlock Reset input is not used, or is otherwise NA, then resets should happen when the input goes from ON to OFF.

Manual Reset: If the WorkPlace Tech Interlock Reset input is used, then resets should happen when the reset is ON. Unlike the automatic reset mode, input transitions from ON to OFF are ignored.

Interstage Delay

The Interstage Delay class of blocks come in three sizes (3, 6, and 10). The implementation currently does not perform a check for out-of-sequence detection. The built-in dual minimum is provided by the dual minimum WorkPlace Tech function block implementation.

Latch

The Latch block has two implementations for the two modes.

Digital Latch: When the Latch and Data inputs are tied to the same value, the Latch block should behave as a Digital Latch. In this case, the entire block is replaced with a SR (Set-Reset Flip-Flop) block.

Sample and Hold: When the Latch and Data inputs are tied to different values, the Latch block should behave as a Sample and Hold block. This usage is different enough from the built-in SHR block that the implementation acts as a wrapper around this block. The SHR will sample Data when Latch is active, and will sample a 0 when Reset is active. If Latch and Reset are both active, the Reset is ignored.

Limit

Limit Thermostat

Loop Sequenced

This block is not yet supported.

Loop Single

This block attempts to use a PIDA block as its core, and map the WorkPlace Tech inputs to the inputs for the PIDA block.

Inputs

Loop Single	**PIDA**	**Notes**
LpEnb	Mod	LpEnb TRUE sets Mod to 1, 3 otherwise
Input	MV	
Setpt	SP	
TR	G	Formula G = N / TR used, where N is a configurable parameter (default 10)
Igain	Ti	Ti = Igain /60
Derv	Td	Td = Derv / 60

OutRef		Unsupported
Action	G	If reverse acting, use -G
RmpTm		Used in post-output ramp
	DZ	Deadzone is configurable as a parameter
	TSg	Feedback loop from post-ramp output

Low Select

Minimum Off

The Minimum Off block behaves the same way as it does in WorkPlace Tech.

Minimum On

The Minimum On block behaves the same way as it does in WorkPlace Tech.

Off Delay

The On Delay block behaves the same way as it does in WorkPlace Tech.

On Delay

The On Delay block behaves the same way as it does in WorkPlace Tech.

OSS

Priority Input

Priority Value Select

Ramp

The Ramp block behaves the same way as it does in WorkPlace Tech. Although the WorkPlace Tech documentation doesn't specify the behavior, testing showed the following properties and were replicated in the Function Block implementation:

- Changing the OutMax value to be less than the current value sets the output to OutMax during the next cycle
- Changing the OutMin value to be greater than the current value sets the output to OutMin during the next cycle

Reset

Schedule 7-Day

Select

Sequence

Setpoint Control

SR Flip-Flop

Thermostat

Thermostat2

Math and Logic

Many of the Math blocks in WorkPlace Tech use NA values to change the way the block works. Unfortunately, this is done in a way where a default value would not be sufficient.

WorkStation Import

During conversion, Inputs and Outputs generate both Function Block IOs and virtual WorkStation inputs and outputs. These virtual inputs and outputs must be bound to physical points, which is dependent on the controller. For each WorkStation input and output generated from the conversion, a list of compatible physical point types is generated in order or preference. During the WorkStation Import phase, the user can select a controller capable of handling the point count and types. The import generator allocates the physical points to the virtual WorkStation inputs and outputs, and generates an XML file that can be imported into WorkStation.