Herman Automation

We boost your efficiency.

# ABB AC450 translation to AC800M code by easy web-tool that saves you time

**Ampl2m** conversion tool was developed during real migration projects to simplify migration of ABB Advant Master AC400 family controllers. AC450, AC410, MP200, AC110, AC160, AC70, AC80, APC are supported.

Manually translating AC450 applications to AC800M can be very challenging and time-consuming. Manually translating a single Controller can take 6 months or even more. Typical challenges of AC450 translation are:

- AC450 Control application is usually quite complex containing thousands of pages of logic
- Control programs may be written in complex way that contains undocumented so-called "hidden terminals" of database elements, which makes it difficult to understand the logic and is therefore difficult to translate
- Programs using MMC-IND or GENxxx PC elements
- Programs using special libraries like RMC/RPC, APC, QCS/1190
- Programs utilizing Type Circuits or User defined PC elements (UDPCE)
- Programs written deliberately in a tricky way, e.g. if the logic depends on the execution order and timing of reading and writing DB elements
- Logic containing DB terminals not available in AC800M, like :SELECTED
- Sequence part translation to SFC requires extensive experience to prevent bad surprises during commissioning

The Ampl2m tool cannot do a 100% complete AC450 conversion, but it can significantly reduce engineering efforts and prevent human errors. The Conversion tool can save up to 95%+ of working hours, based on experience. The Unconverted signals and functions, shown in red in AC800M code, should be fixed manually.

Based on author's experience in many projects already commissioned, it is really worth using the conversion tool to reduce all that exhaustive manual migration.

## Core Benefits

- Automated conversion can save hundreds of working hours per each Advant CPU
- Conversion tool is easy-to-use and available on-line 24/7
- Prevents human errors
- Conversion process is interactive thus under your control
- Pulp & Paper Library / Standard AC800M libraries / UserLib / any libraries may be selected for conversion
- Any library can be used in the conversion by instantiating unfolded Diagram Templates, see Step3 conversion
- Converter utilizes hundreds of SW solutions integrated into one common platform, verified in successful projects and developed based on a long-term experience in AC450 & AC800M programming
- Converted code is efficient in terms of CPU load, easy to understand and free from any complex nested structures
- Conversion tool can handle even very complex applications
- Cost-saving in terms of tag count
- Original logic can be converted to CBM Programs, Single Control Modules or Control Diagrams

- Diagram Templates can be utilized in converted Diagrams, either folded or unfolded, similar like Type Circuits in Advant systems

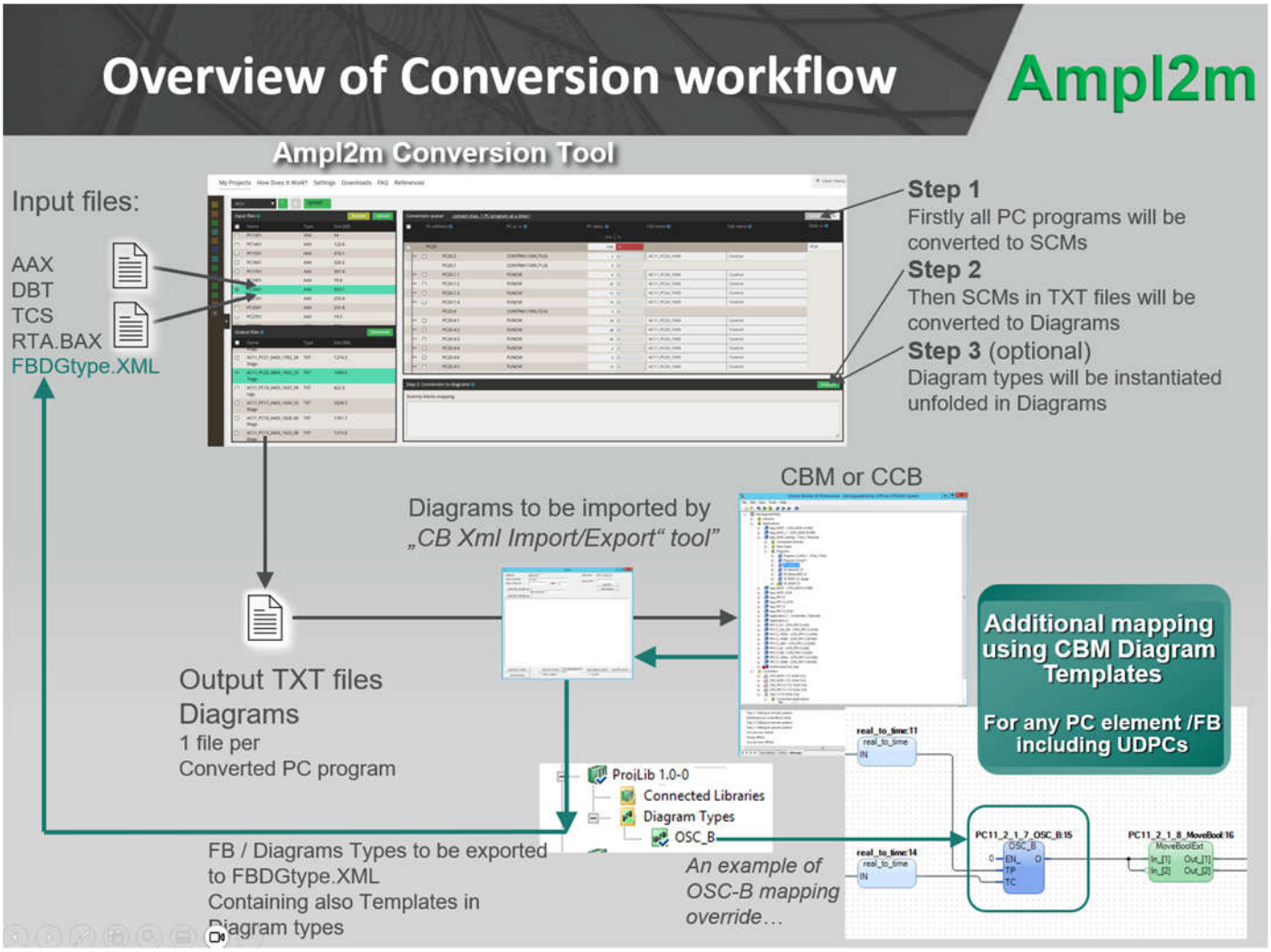
## How does it work?

Conversion tool is built as a database application hosted on a Virtual private server (VPS) . It is available as a web service at **ampl2m.com** and accessible through a simple web interface, ready for instant conversion. Just log in, upload source files and press Analyze and Convert button. Converted programs are available for download within minutes.


Note: Max. 1 PC program shall be converted at a time.

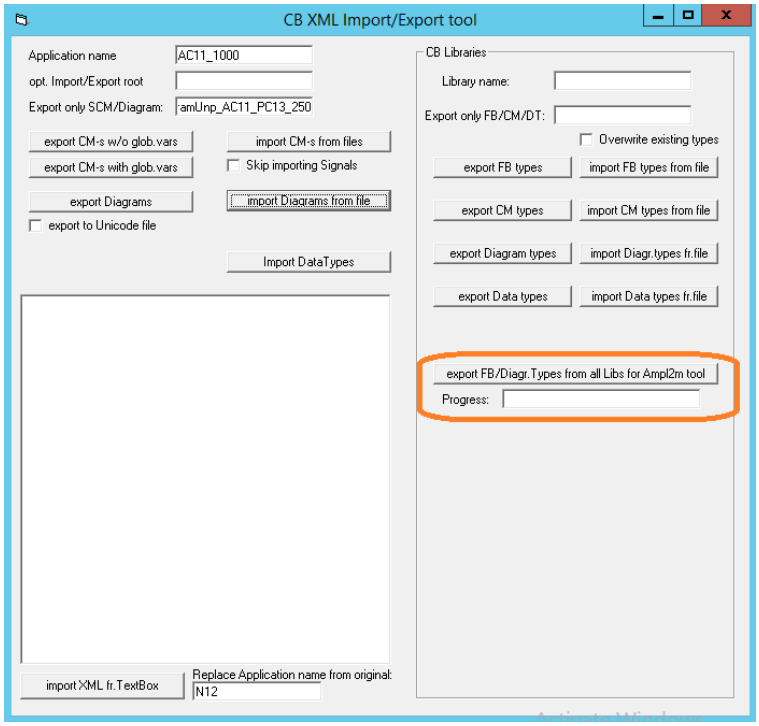
The output text files, available for download in “Output files” section, contain all the logic, variables, data types and constants ready for import into 800xA CBM or Compact Control Builder. The import is done using a short VB script using the ABB Open interface of Control Builder.

Conversion is subject to payment or the conversion tool can be used in demo mode, producing only a few pages of converted logic.



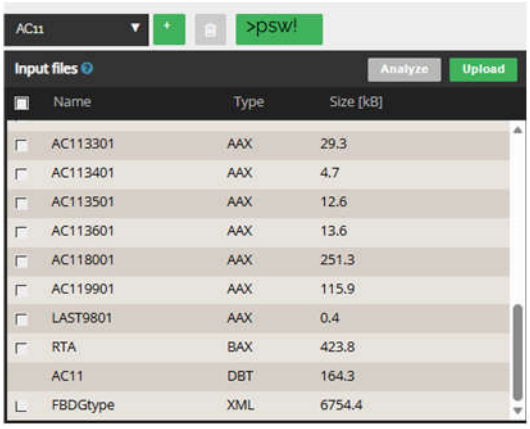
## Input files

- **AAX files** to be exported from Function Chart Builder (FCB). AAX names should be in format generated by FCB, e.g. AC112201.AAX , it means PC program PC22 of node AC11.
- **DBT file** is a “Export DB Section” file from FCB containing all DB elements including all their parameters. Only 1 DBT file can be uploaded for one PLC containing all DB Elements. Make sure “Include Headline with DB Terminal Names” is checked (by default).
- **FBDGtype.XML** file is needed just for a conversion to Diagrams. It should contain export of all necessary libraries linked to the Application to be used as a destination of the converted logic in Control Builder (CBM). FBDGtype.XML can be exported from CBM by means of “CB XML Import/Export tool” , a small exe application to be used in the engineering station of 800xA along with CBM.




- **RTA.BAX** (optional) is a database backup file from existing RTA board on site. If RTA backup is uploaded, the conversion tool creates automatically RTA init Control module in the first page of each Diagram. RTA Init CM initializes RTA structure containing all Treatment numbers used in the converted AC450. Treatments are linked automatically to Alarm/Event definitions in Diagram Templates.
- **TCS files** (optional) contain Type circuits (Templates or Typicals) to be excluded from converted logic and to be replaced by one block (Function block or Diagram Type)

Example of Input files section:



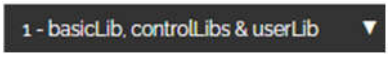

## Output files

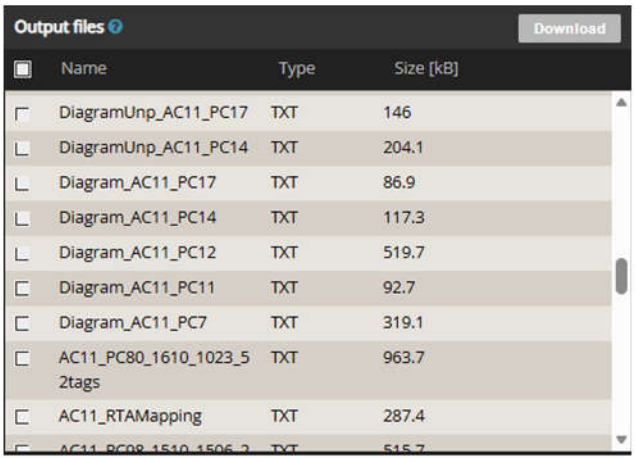
All Output files are generated by the conversion with extension .TXT.

- **Step1 converion** creates TXT files containing **Single Control Modules (SCM)**. Example of the file name: AC11_PC14_1510_1052_12tags.TXT, which means PC14 has been converted, it can contain several SCMs for different cycle times, conversion done on 15th October at 10:52. The file contains 12 objects with names.
    - SCMs can be imported to CBM as SCMs or as Programs according to selection in the import script
    - No need to import SCMs to CBM in case of conversion to Diagrams. SCMs work as an intermediate step for conversion to Diagrams in Step2.
- **Step2 conversion** from SCM to CBM Diagrams creates file names like Diagram_AC11_PC12.TXT. One TXT file contains all Diagrams of different cycle times of 1 PC program, in this example PC12.

  If any of the **Diagram Template** included in FBDGtype.XML file match PC element call-names, these Diagram Templates will automatically override predefined mapping in the conversion tool and be used instead of the mapped Function blocks of PP Lib/UserLib. The names of the Input/Output parameters must match the PC element Input/Output names. Dash "-" character in PC element's call name or in Inputs/Outputs to be replaced with Underline "_" characted in parameter names of Diagram Templates.

  Diagram Templates work in converted Diagrams in a similar way to Type Circuits (TCS) in Advant systems.
- **Step3 conversion** is used to unfold **Diagram Templates** (DT) so that the internal logic of the Diagram Template is inserted/instantiated in the Diagram in place of the original instances of Diagram Template. A Diagram Template can contain multiple pages, but it is recommended to limit internal logic to only 1 page.
    - DT to be unfolded must contain a local **bool variable** named "**Unpack**" with **Initial value=true**
    - Name of DT (e.g. DOC) to be used in the DT's inner logic as prefix in names of Parameters, Variables, Control modules, Function Blocks or even Diagram instances inside Diagram Template, so that the DT name will be replaced with current object name linked to the parameter Name of DT.
      For example: "DOC_IPar" name of structured variable will be instantiated as "[ObjectName]_IPar" in Diagram, etc…
    - For Step3 conversion, it is necessary to use Library selection "1" in the Settings section

      
      
    - Example of the file generated by Step3 conversion: DiagramUnp_AC11_PC12.TXT




## Security, Privacy

We are committed to protect your data. We take the protection of your personal data very seriously. We are using SSL encrypted connection (https:), so third parties do not have access during transmission of data between your browser and our Virtual Private Server (VPS). You can recognize an encrypted connection in your browser's address line when it changes from http:// to https:// and the lock icon is displayed in your browser's address bar.

Our VPS is located in the Data center of VPS Provider in Germany secured by DDoS protection and Firewall. Your Input and Outputs files are stored inside the database in our VPS. External connection from internet to our database is disabled, therefore the database can be accessed by Conversion service only . Remote access to VPS is secured by SSL Private Key thus restricted to Herman-Automation only.

Herman-Automation does not share with third-party nor sell to third-party Your personal data or Your files. Read our Terms&Conditions for further information.

## What is the efficiency of Ampl2m conversion tool ?
## How much manual work is needed after automated conversion ?

Automatic conversion can save up to up to 95% of manual AC450 translation efforts, depends on how original code is written. If original control application is built using standard functions then automated conversion covers almost all engineering efforts regarding Logic conversion.

According to opinion of many ABB specialists complete AC450 to AC800M automated conversion is not possible due to significant difference with application handling between AC400 and AC800M. Manual post-conversion job and experience are important parts of the conversion process.

Amount of necessary manual programming after automated conversion depends on the way how original code is written:

- Automated conversion produces nearly working AC800M code if original logic uses standard features of AC450 function blocks and DB elements – in case standard MOTCON, VALVECON, PIDCON, MANSTN, SEQ are used

- On other hand converted code needs manual fixes proportional to amount of GENUSD, GENCON, GENBIN, MMCX used and on how sophistically and tricky is original code written.

In our experience, there is enough room for partial automated conversion which can significantly reduce engineering hours and prevent typos, mistakes due to inattention or logic errors. **Manual completion after automated AC400 conversion may take 2-6 weeks, depending on experience. That manual post-conversion job is much easier than at least half-a-year-long manual reprogramming of AC400 to AC800M.**

You can send us AAX, BAX files for assessment how many working hours will be needed in your project for manual fixes in converted code.

## Features

- Input files required for automatic conversion: *.AAX , DBT (Advant controller database export file), *.TCS in case AAX files contain Type circuits. Note: BAX file is not required hence it secures AAX files cannot be opened in Function Chart Builder by other person.

- Output TXT file contains translated all global variables, constants and Single Control Modules (PC programs) ready to import into 800xA / CCB. Typically one output TXT file contains one translated PC program.

- Advant AMPL code is being converted into Programs or Single control modules(SCM) using FBD or SFC language in Step1 conversion. Selection between Programs or SCM is available in the Header of the VBS Import script for importing to CBM.

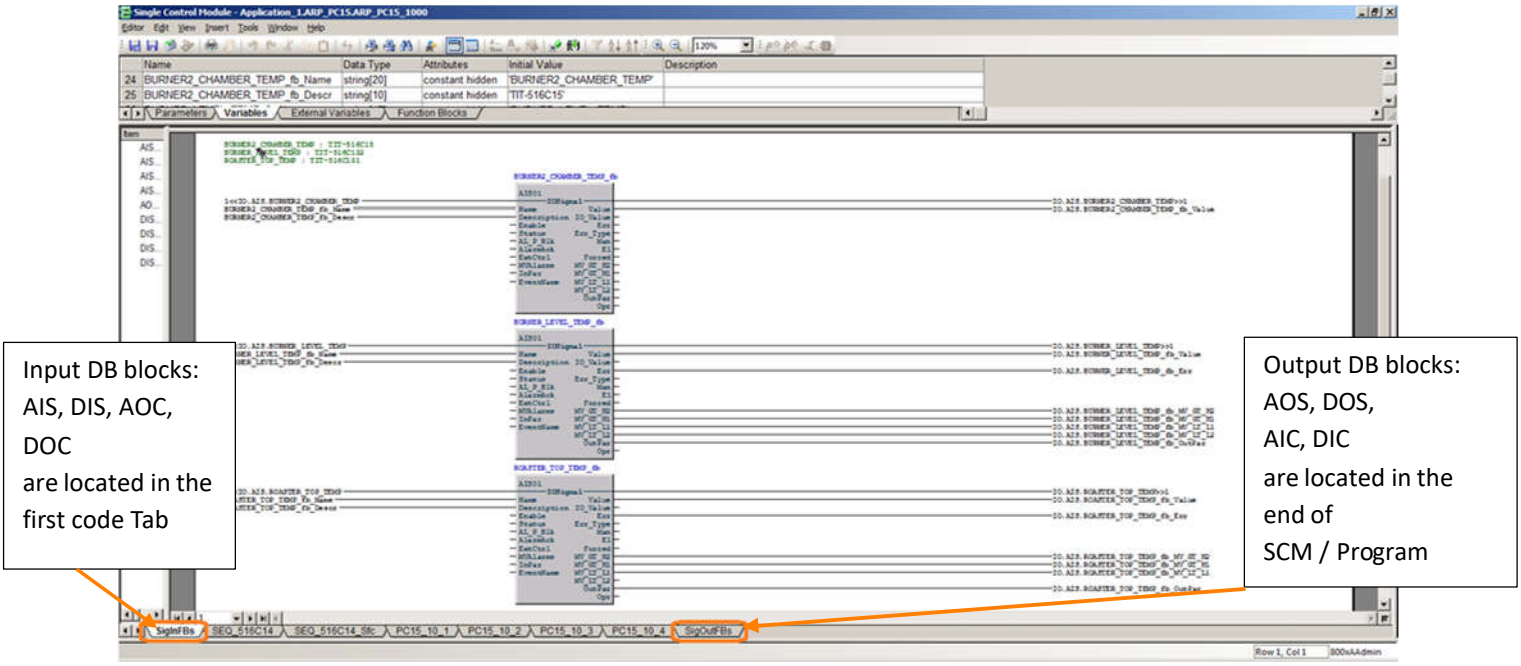
- Single Control Modules (SCMs) are being converted to CBM Diagrams in Step2 conversion.

- **IO function blocks (represent DB blocks AIS, AIC, DIS, DIC, AOS, AOC, DOS, DOC)** are distributed in converted Programs close to the related logic.

  There is built-in SQL logic, which decides for the most appropriate location of each single IO function block in Single control modules (PC programs), according following rules in descending priority:

  - The fastest program cycle is selected with the connection to the particular DB / IO element
  - IO function block VALUE or CALC_VAL is written
  - IO function block VALUE or other terminals are red the most times in particular PC program
  - These AIS and DIS, which are not connected in any PC program, their IO function blocks are created in dummy PC99 generated by conversion tool. That dummy PC99 should be converted as the last one. The most likely AIS and DIS collected in dummy PC99 are used as independent measurements just displayed in HMI graphics.

  Local distribution of IO function blocks in PROGRAMs is very helpful for easy tracing logic and for navigation from HMI / Alarm list / HW IO cards to the logic. Example of how IO function blocks are created at the first and the last tab of each PROGRAM:




- RealIO and BoolIO variables are created locally in the same SCM/Program/Diagram along with corresponding IO function block in case of Local preference, or globally in case of global preference. IO variables are furnished with original IO name, Range, Unit and Fraction in variable description.

- Name and Description texts are created as local “constant hidden” strings in SCM/Program (by default, as per example above) or as project constants organized in folders of constants like cNames.<Application name>.<Const name>. The same is applied for Names and Descriptions of Motcons, Valvecons, Pidcons, Manstns and SEQ function blocks...

  Name and Description constants are uploaded automatically to the Control Builder during importing of converted code.

  Names and Descriptions will be connected directly to the input parameters in Diagrams

- Extensions are being added in the end of IO FB names in order to prevent collision between FB name and RealIO/BoolIO variable name.

  Name extension of IO FB is adjustable in the converter’s User settings. Default extension is “_fb”. If extension setting is preset to “_#” then particular type of IO function block is used as extension, e.g. <DB name>_AIS , <DB name>_DOC ,…

- IO values are referenced in the logic in 2 ways:

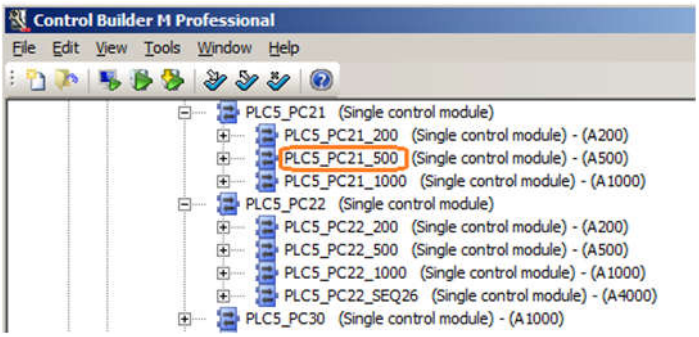
  - Locally by <IO FB name>.<terminal> within the same PROGRAM where IO FB is located

  - by global variables connected to the terminals of IO FB, if IO value is used also in other Program (s). See chapter “Preference for local/global variables” below.

- Original Descriptions of IO function blocks are also added in Page comment in which particular IO FB is located and also in the remark above FB in process logic reading values from IO FB

- If Local variable preference is preselected before conversion then RealIO / BoolIO variables (to be connected to IO cards) are created locally in the same Program in which particular IO FB is created.

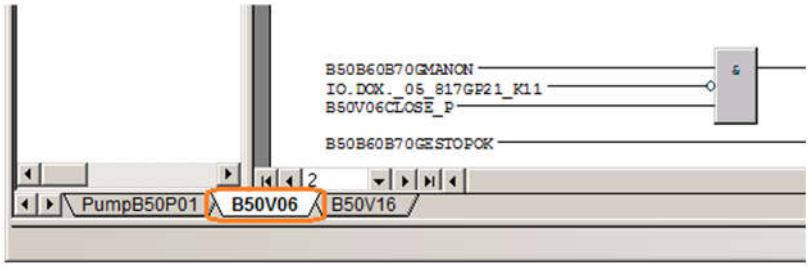
- **Code distribution** can be adjusted by setting appropriate names of target Programs / SCM and code tabs. If the same name is set for several code blocks (FUNCMs), their code is merged into one code tab. This approach can decrease AC800M CPU load significantly, especially if there are a lot of small FUNCM/SLAVEM code blocks in AC450.




○

Programs / SCM names can be modified before conversion. Programs can be split/merged by setting new/another existing Program name instead of a default name preset by Analyze function.

Analyze function presets Tab names by default as original PC addresses of FUNCMs like PC22_3_2_6. Modify Tab names according particular control function or plant part. Merge tabs if possible before conversion by using the same Tab name for several original FUNCM code blocks.




- **Preference for local / global variables.**

  In case Local variables preference is set, global variables are created only if particular variable is used in more then one Control module, mainly in case of communication between PROGRAM of different cyclicity :

  This preference is applied for named connections in the logic and for connection between IO function blocks and the logic. Local IO signals are referenced by <IO FB instance name>.<terminal> .

  **Local or global preference - what is the best option?**

  Global variables preference is handy because all variables are accessible across all converted PC programs.

  On other hand, amount of global variables is limited within one Application (~65536). Unfortunately Control Builder complains if maximal limit is exceeded just on download to the real AC800M CPU. This problem may be discovered too late if CPU is not available during programming phase of the project.

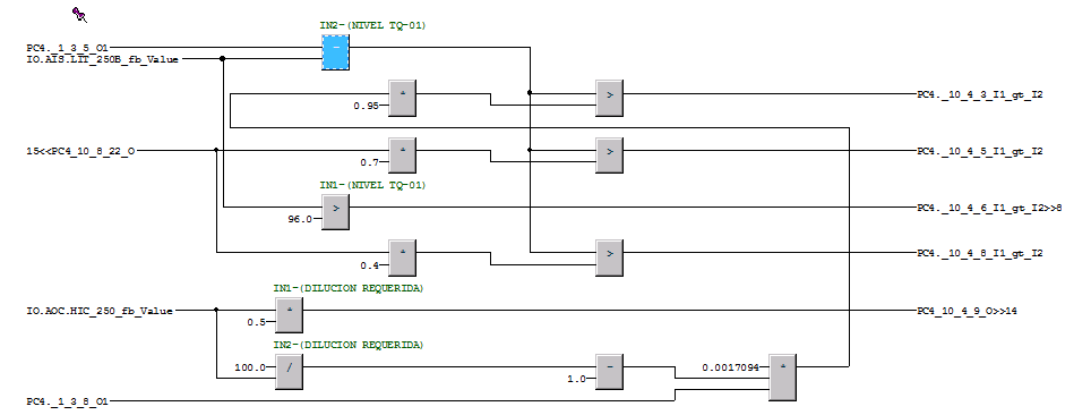
  **Therefore, the safest option is to prefer Local variables, especially when convering to Diagrams or if the control logic is complex – say, in the case of more than 20 PC programs per one CPU.**

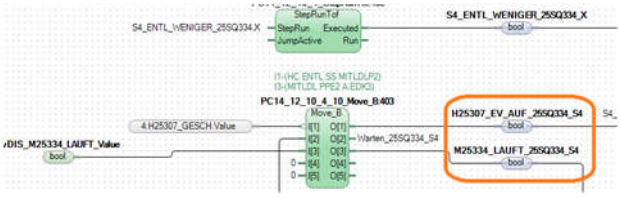
- **Time entries** (like D=1:0:0) can be converted as time constants (cTimes._1h) or coldretain time variables if adjustment of time entries is expected.

  All time constants used are listed in the end of output TXT file and they are automatically added as project constants during importing converted code into 800xA. The same applies for String constants.

- **Symbolic names translation.** Original symbolic names at PC element outputs are used as names of new variables connected at converted function blocks outputs.
    - If original symbolic name is enclosed in parentheses, symbolic name may contain even special characters. In that case conversion tool converts special characters as abbreviations, e.g. “<” is replaced by “_lt_” , “%” is replaced by “perc”...
    - In case there is no symbolic name available for the particular PC element output, following rules are applied:
        - In case of objects (Motcon, Valvecon, Pidcon, Manstn, AIS, DIS,...) , output variable name is created as <FB name>_<terminal name>
        - if output variable is connected in other Single Control Modules, then global variable is created like
          <PC prg.>.<remaining part of original PC address>_<PC element output terminal>
          e.g. PC15._1_6_O
        - if output variable is connected to the neighbour code tabs within one Single Control Module, then named variable is created like
          < original PC address>_<PC element output terminal>
          e.g. PC15_1_6_O
        - if output variable is connected inside particular Code Tab only, then variable name begins with 2 underline characters – these variables are not listed in variable list of Control Module
    - **In case symbolic names are lost** in uploaded .AAX files from Advant controller , it is possible to restore the original symbolic names during converting. In that case, ask customer for older AAX/AA files that still contain symbolic names. Upload these old AAX files to the conversion tool with names starting with „SYMB“ , e.g. SYMB1201.AAX . The old symbolic names will be extracted from the SYMBxxxx.AAX files and used in the converted logic. This automatic recovery of symbolic names has already been successfully used in the real projects.

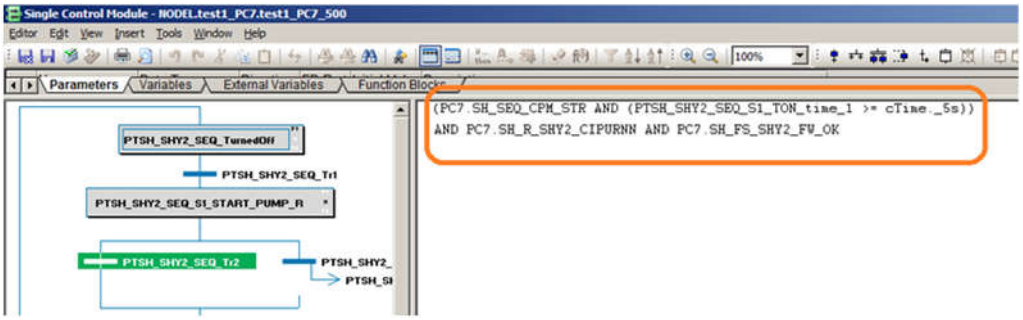
Example of alternate variable naming if original symbolic names are not available:



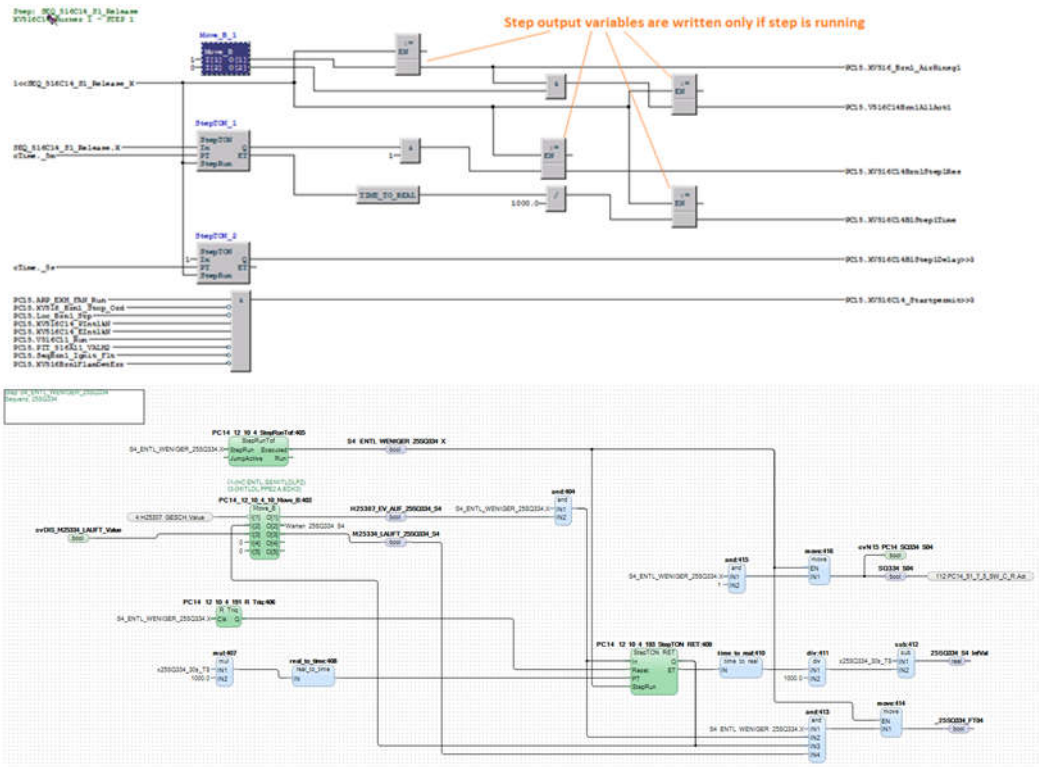
- **Selection of STEP transition variable:**
    - single bool variable
    - whole logic expression.

    What can be helpful for more informative HMI in SFC Viewer (available only if whole STEP logic is converted into SFC only).



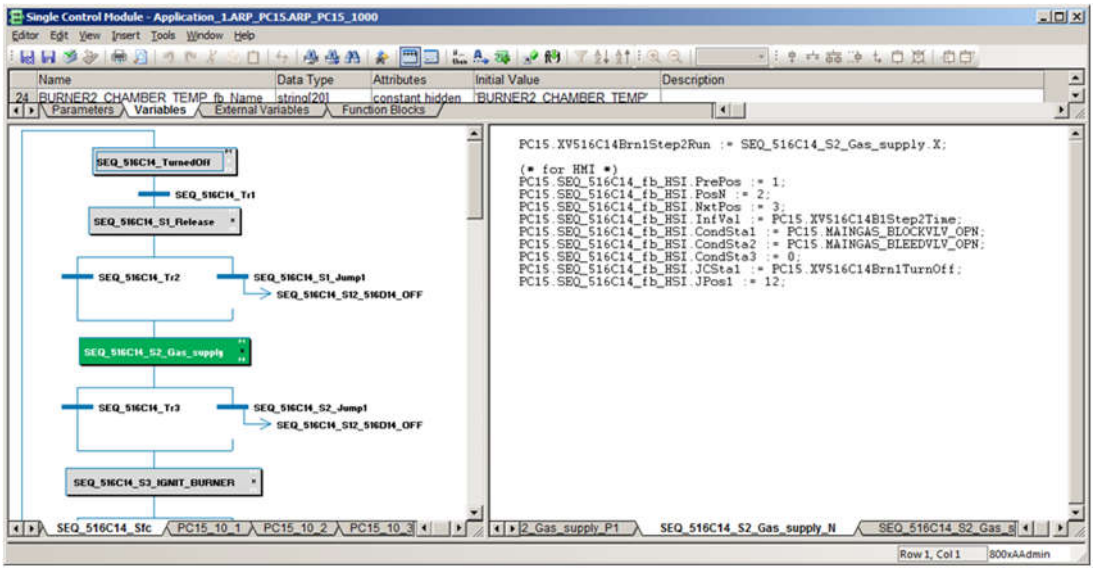

- **SEQ conversion:**
    - Whole STEP logic is converted into SFC as by default. This is the best option if Steps contain just one page of simple logic like few timers and MOVE sending commands to the valves, motors and PIDs.
    - In case STEP logic is complex, for better readability it is useful to convert STEP logic into FBD code located one tab before SFC. In this case SFC is used as a „SEQ skeleton“ for enabling particular FBD Step logic.

    Step code examples, if FBD is selected in User settings for SEQ translation:



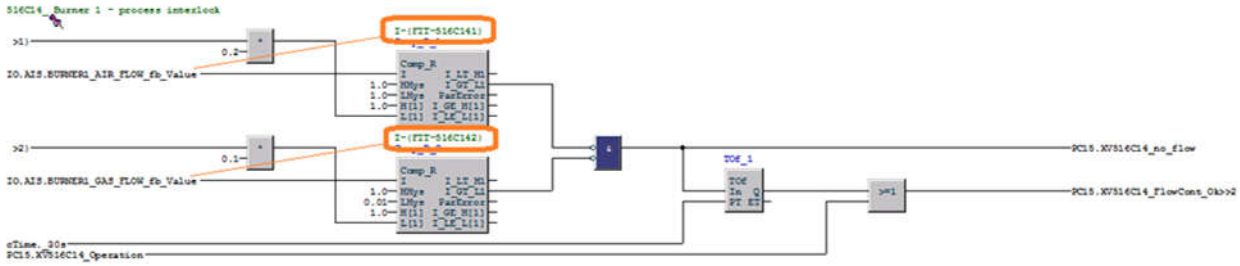
- SFC contains Steps (without process logic), Step Timers, Transitions, Jumps and live values to SEQ Faceplate
- JPOS jumping from any to any Step is supported
- SFC contains logic for showing live Conditions with texts , Info Val, etc...

SFC skeleton example:

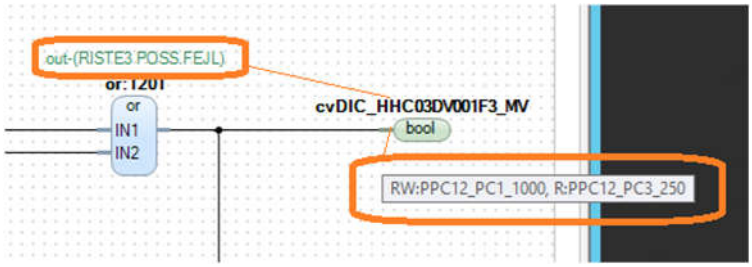



- **Number of 800xA Tags can be optimized** by means of User filters of IO signals which do not need faceplates. The aim of tag filters is to avoid using IO FBs for motor's MCC signals and limit switches of valves, which values are being displayed in faceplates of corresponding valves and motors already.

- **Description of connected IO signals** is added into the remark above function blocks.

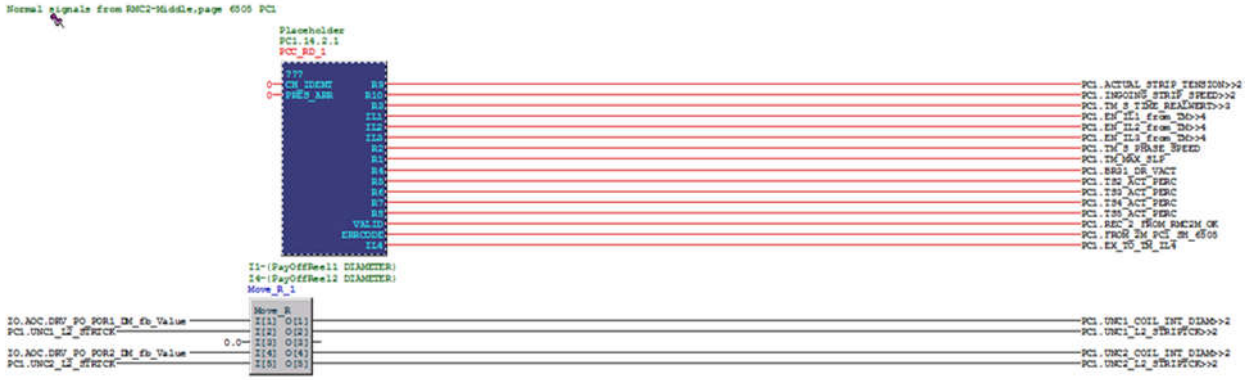



Another example of DIC Description



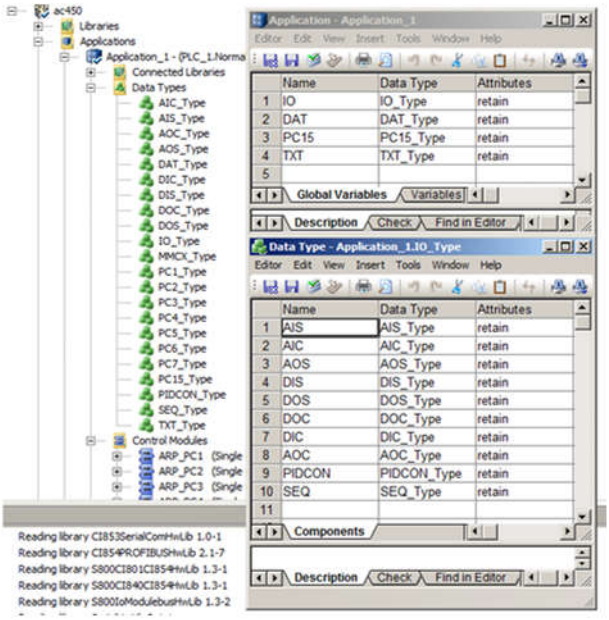

CV quick cross-references

- In case of rarely used PC elements, which are not mapped in the conversion tool yet, they are being replaced by red **placeholders**. At the same time all input / output logic connections are translated and maintained. Names of placeholders are added to the FunctionBlocks definition tab. Just a new FB definition in User Lib left to be added manually.



  - if this unmapped PC element contains connection to DB element, DB name is used to name an instance of the placeholder, in this case Name and Description parameters are filled automatically
  - All Placeholders are listed in the end of the output TXT file.

- **Data type structure** prepared by automated conversion is shown in the following screenshot. Data types are created in Control Builder during importing of TXT files, which contain Control modules and data types. During sequential importing of several TXT files (since 1 TXT file contains 1 PC program usually) variables are added into the existing data types imported before. No variables previously created are deleted during importing into Control Builder.
  - Description of every variable contains information where particular variable is written from – source PROGRAM name and Tab name.
  - Unit is also placed in variable description.
  - Initial value is set for time and string constants and for numerical constants used multiply in the logic (MD=… )
  - All original int variables are converted as dint



- **Tag counter is built in**. Tag quantity is shown in the name of each output TXT file.
- **Type Circuits.** Conversion tool can translate Type Circuits in case:
    - all TCS source files are available and uploaded at your Input files before conversion
    - AAX files contain patterns like „(* Begin of Type Circuit“ -and- „(* End of Type Circuit“

```
(* Begin of Type Circuit PC89.1.1.1    INTFSV2                *)
(* End   of Type Circuit PC89.1.1.1    INTFSV2                *)
```

        - OR

    - TCS file header contains a regural expression list that defines significant parts of TCS that must match to AAX code. In this case Type circuits are being identified in AAX code automatically even without gates „(* Begin of Type Circuit“ , „(* End of Type Circuit“ in AAX.

```
(*****************************************************************************)
(* ME.TCS generated with FCB Release 6.0/0              30-MAR-2021  13:06:39 *)
(* Pagelayout: DIN 6771  A4 landscape  English            Rows: 90, Cols: 185 *)
(* Node type: AC 450 with SW QC07-BAS41* 4.0          Base SW: QC07-BAS41 4.0 *)
(* Additional options: QC07-COM41 4.0, QC07-FUZ41 4.0, QC07-LIB41 4.0,        *)
(* QC07-LIB42 4.0                                                             *)
(* ~\n(PC\d+[^\s]+)\s+BLOCK\s@\n$\.100\s+MOTCON\s*\(@\n$\.2\s+MOVE\s*\(B,5@!\s$\.100:SO2#  *)
(*****************************************************************************)
```

    - Type circuits are translated as Function blocks instances in FBD code. Note that Function block types should be prepared manually in the current version of the Conversion tool.
    - “extra” PC elements inserted within boundaries of TC in AAX are converted after FB

- **IO variables connection to IO cards.** CSV file (comma separated text) is created during importing of translated logic to Control Builder M. CSV file contains list of BoolIO, RealIO variables used in the converted logic including DB element original names, Full path of IO variables, Analog ranges, units, fractions and channel inversion. CSV file data can be easily copy-pasted to the IO card editor. We cannot generate complete HW structure of IO cards since we don’t know your new IO cards layout and IO signals distribution in IO cards.

## Libraries

**PP Library** (Pulp & Paper) is used in converted code as the first option (default). If PP Library is available and/or requested by customer it is advantageous for AC450 conversion because:

- PP function blocks are compatible with Advant controller’s function blocks. Also Ampl2m conversion tool is very well optimized for using PP Lib functions.
- PP HMI Faceplates have similar functionality as control objects at AS500 operator stations and nearly the same look and feel like 800xA+AC400connect HMI.

**UserLib**

**Reliable and cost-saving solution by means of using Standard AC800M libraries + UserLib**. In this case FBs are used from BasicLib, ControlSimpleLib mostly. In addition, there are function blocks created in UserLib replacing AC450/APC

FBs which are more complex hence not easily convertible by standard library FBs. UserLib FBs were tested thoroughly and used in the real projects already. UserLib is password free hence open for modifications.

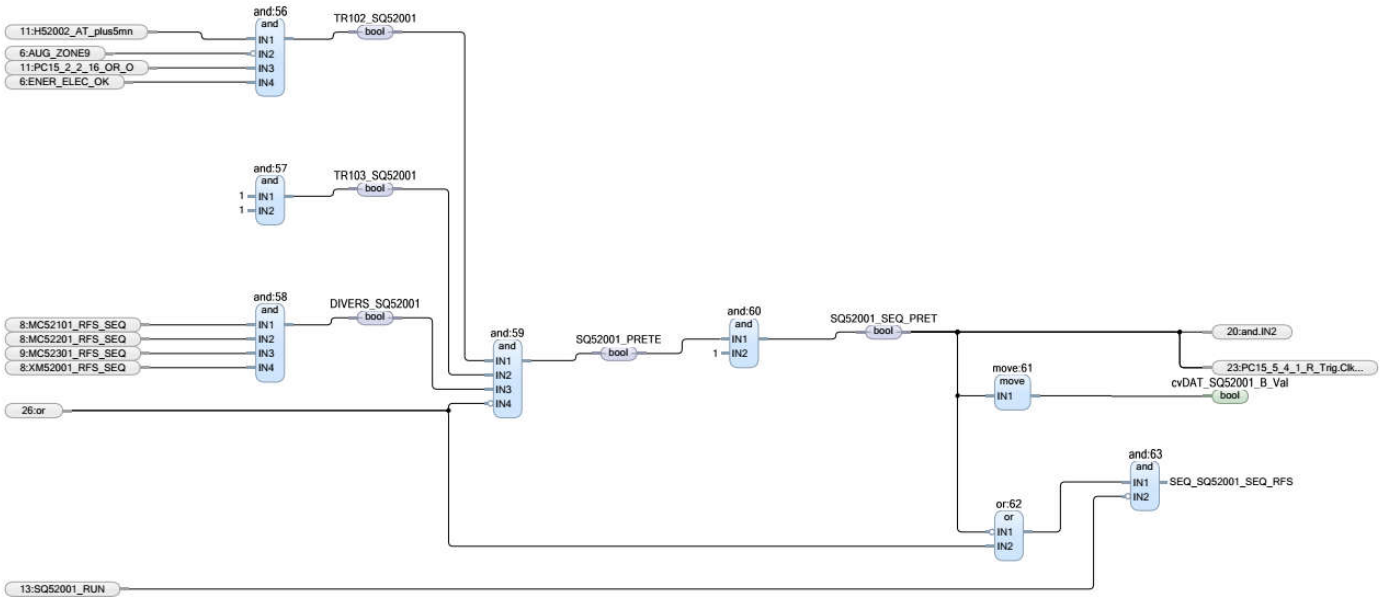
See how Libraries are utilized in the list of supported function blocks in the end of this document.

**Any library** can be utilized in the conversion code by means of Diagram Templates prepared in CBM. During the conversion Step3, the entire contents of the Diagram Templates are unpacked into a Diagram, including all variables, functions, FBs, CMs used in the Diagram Templates.

## Step2 - Conversion to Control Diagrams

Example of converted logic from Output TXT files (FBD code) to Control Diagrams. Conversion tool calculates the position of each single function block in a logical chart in a similar way as FCB does.

Communication variables are used instead of global variables.




## Testing of converted code

It is helpful, but not mandatory, to test the converted code in simulation mode before commissioning. According to experience, the testing of the converted logic helps to get familiar with the logic thus it makes commissioning easier. All motor MCC feedbacks and limit switches of valves should be simulated in AC800M while IO cards are disconnected. The aim of tests is to perform simulated start and stop of controlled plant. It is recommended to run all sequences and check if all particular motors, valves and PID controls perform as expected.

## Conversion tool released version

Ampl2m conversion tool has been tested by several experts in ABB control systems. It has been used in many migration projects successfully. The tool has been used by the author in X projects including commissioning.

Ampl2m conversion tool development is in progress with the aim of increasing its efficiency. Meantime the conversion tool is ready to help in migration projects as solid stabile version. Any bug reported will be fixed ASAP.

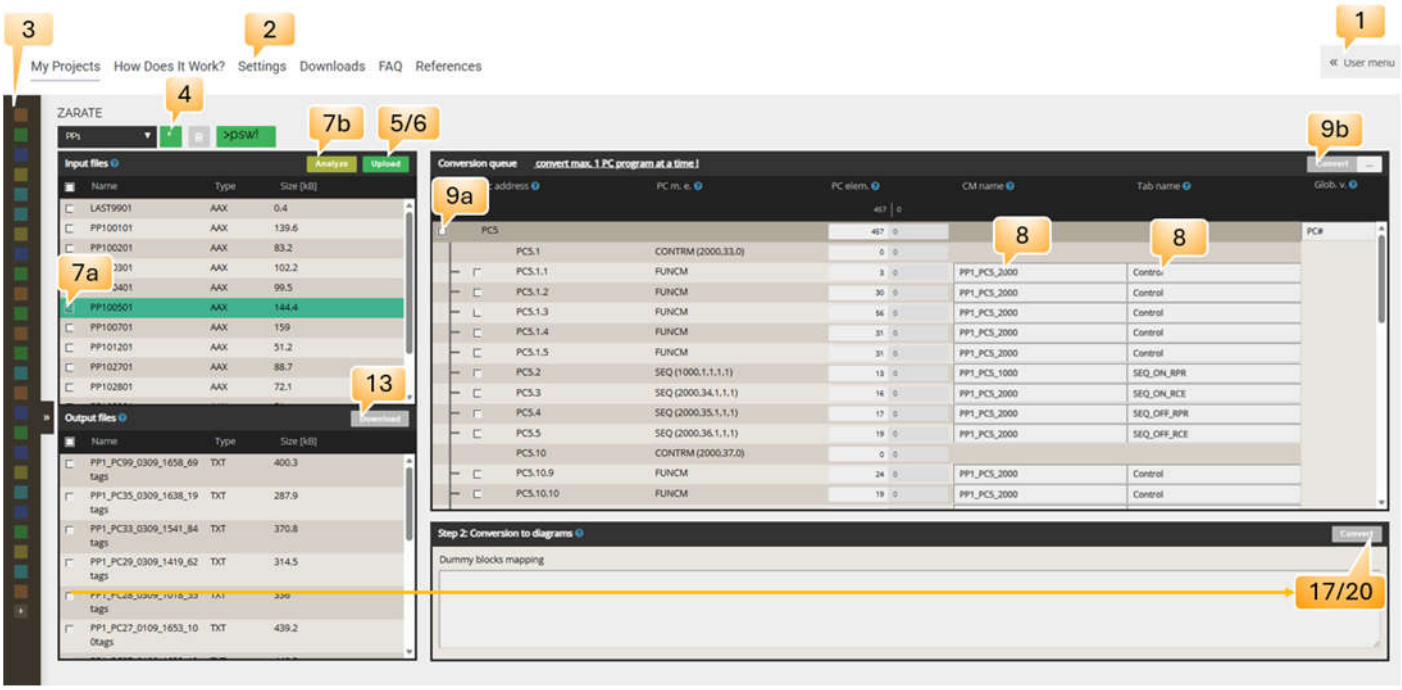
Punch lists or any idea for tool improvement are appreciated much. It is possible to adapt the conversion tool for your project specific needs or for other libraries, see “List of supported PC elements” chapter.

## Guarantee of Service quality, Warranty

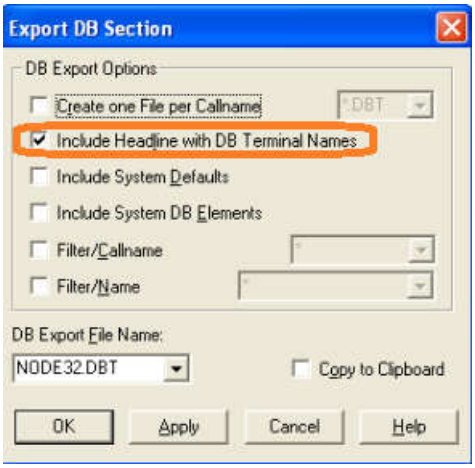
Herman-Automation warrants that, for any paid Conversion , for PC elements supported only, the Conversion will produce control logic equivalent to the original control logic from Input files (Equivalent logic means that converted logic is producing the same outputs as original logic if both control logics are working with the same inputs).

The warranty does not apply to the free Conversion.

## Automated conversion Workflow




1. **Create your account** at Converter web site
2. Check and adjust your **preferred settings** e.g. select Libraries to be used, Page layout, local/globar preference, code optimization settings...
3. **Create new Project**
4. **Create new PLC.** PLC name must be unique within one Project. You can use PLC name from your Advant engineering PC FCB.
5. **Upload PC source files** *.AAX and DB export file *.DBT from FCB (use File -> Export DB section… and confirm default settings). Only one DBT files should be uploaded containing the export of the whole DB part. All AAX files should be uploaded prior to the first conversion of particular PLC.
   How to export DB Section into DBT file in FCB:




6. **Upload FBDGtype.XML** file exported from Control Builder if you plan to convert logic to Diagrams.
7. **Select AAX file(s) for analysis and click Analyse button.** All AAX files can be selected and analyzed at once. Converter parses selected AAX files and calculates number of convertable function blocks.
8. **Check and adjust suggested Control Module names and Tab names** in Conversion queue table. It is highly recommended to change Tab names to appropriate names like plant part abbreviation. Merge

several code blocks by means of entering the same Tab name. Rule of thumb: Less Tabs = lower AC800M CPU load and faster opening/saving of CM editor.

9. **Convert:** Select code blocks in Conversion queue table and Click Convert button. It is recommended to convert one PC program at a time (one output TXT file per one PC program). Do not convert only part of PC program because some variables may remain unconnected. Alternatively click on PC program header line in Conversion queue to fold/unfold PC prg.
   Convert automatically generated file LAST9901.AAX as the last one. That file contains all IO function blocks, which are not connected in PC programs. (typically AIS, DIS used in HMI only)
   You can repeat conversion of a current PC program but **never convert to previously converted programs** otherwise distribution of DB elements in PC programs may change and duplicates of some DB elements may occure.
10. **Select** limited conversion for free or full conversion if you would like to (demanded). This selection has no meaning meantime since conversion tool is free.
11. After selection of the full or demo conversion, popup "Conversion status" appears and displays the progress of conversion.
12. Conversion takes from several seconds to several minutes depending of AAX Advant code complexity

If you convert to Diagrams, bypass steps 13-16 and continue with step 17.

13. Select first and then **Download converted text file**, which contains all converted Control Modules and data types in text XML format ready to import to your 800xA system.
14. Adjust selections in the header of the Import VBS script.
15. **Import converted text file** to the 800xA system by means of VB script (available in Download section)
16. Define global variables in the Application using imported data types like IO,DAT, AIS, DIS, AIC, AOC,…

Step2 conversion

17. In Output files section, Select one converted TXT file with a name in the form *PlcName*_PC*xx_DDMM_hhmm_xx*tags and click the „Convert“ button located in bottom section „Step2: Conversion to Diagrams“. Repeat conversion of each PC program one by one.
18. Select first and then **Download converted text file** „Diagram_...“, which contains all converted Diagrams for selected PC program

Step3 conversion

19. In Output files section, Select one converted TXT file with a name in the form „Diagram_*PlcName*_PCxx“ and click the „Convert“ button located in bottom section „Step2: Conversion to Diagrams“. Repeat conversion of each PC program one by one.
20. Select first and then **Download converted text file** „DiagramUnp_...“, which contains all converted Diagrams for selected PC program with unfolded Diagram Templates

## Post-conversion Phase 1

is defined as manual programming after automated Conversion, in order to fix all unconverted parts of Output files and to get converted programs ready to download to AC800M controller, including following activities:

- Visual checking of imported code page by page according original logic
- fixing all Placeholders in converted Output (Placeholder replacement by appropriate Function blok of AC800M)

- fixing all red connections between Function blocks. Converted code may contain red connections, because special DB hidden terminals or terminals SELECTED were used in original logic.
- fixing all Code loop errors during compilation in Control Builder M (only if SCM:s are used)
- assigning Tasks to Converted code in Control Builder M
- defining communication between other controllers as replacement of DS and DSP communication blocks
- Downloading project to the Softcontroller

## Post-conversion Phase 2

is defined as configuration in Control Builder/Engineering workplace in Customer's 800xA system in order to:

- to set up Coldretain values into Pulp&Paper function blocks in Control Builder M online
- to build hardware definition of AC800M and all its hardware components
- to setup IO variables to IO cards, in Control Builder M
- to configure texts of Sequence steps in 800xA Engineering workplace
- to configure interlock texts of Valves and Motors in 800xA Engineering workplace
- to download converted code into AC800M CPU and to optimize CPU load and Tasks
- to perform FAT test with Customer / end-User
- Commissioning of converted code

Herman-Automation can take over responsibility for post-conversion activities and can provide Post-conversion Service including whole or part of Post-conversion activities defined above, according Purchase Order from Customer.

## List of supported PC elements

Ampl2m conversion tool supports now almost 100 PC elements, which are the most frequently used in the control logic of Advant controllers. Following table explains how particular Libraries are utilized in the converted applications. Conversion tool is improving and number of supported PC elements is increasing after each project.

The built-in mapping of any PC element can be overritten/extended by using Diagram templates with the same call names as PC element.

| PC element | Selection User Lib | Selection PP Library |
|---|---|---|
| AND | Basic Lib | |
| OR | Basic Lib | |
| TON | Basic Lib | |
| TON-RET | Basic Lib | |
| MONO | Basic Lib | PP Lib |
| MOVE | User_Lib | PP Lib |
| MOVE-A | User_Lib | |
| OR-A | Basic Lib | |
| INV | Basic Lib | |
| STEP | Basic Lib | |
| CONV | Basic Lib | |
| SR | Basic Lib | |
| FUNCM | N/A | N/A |
| TRIGG | Basic Lib | |
| SR-AO | Basic Lib | |
| SW-C | Basic Lib / User_Lib (T) | PP Lib |
| TOFF | Basic Lib | |
| MUL | Basic Lib | |
| SUB | Basic Lib | |
| SR-OO | Basic Lib | |
| COMP-R | User_Lib | PP Lib |
| AND-O | Basic Lib | |
| DIV | Basic Lib | |
| BLOCK | Basic Lib | |
| INT | ControlSimpleLib | PP Lib |
| ADD | Basic Lib | |
| REG-RET | User_Lib | |
| COUNT | Basic Lib | PP Lib |
| CONV-BI | Basic Lib | PP Lib |
| LIM-N | Basic Lib | PP Lib |
| COMP-I | Basic Lib | PP Lib |
| REG | User_Lib | PP Lib / User_Lib |
| MUXA-I | User_Lib | PP Lib |
| THRESH-L | Basic Lib | PP Lib |
| COMP | Basic Lib | |
| OSC-B | Basic Lib | PP Lib |
| SR-AA | Basic Lib | |
| CONV-IB | Basic Lib | PP Lib |
| MUX-N | Basic Lib | PP Lib |
| MUX-MN | Basic Lib, User_Lib | PP Lib / User_Lib |
| COUNT-L | Basic Lib | PP Lib |
| SR-D | User_Lib | PP Lib |

| | | |
|---|---|---|
| MUX-I | Basic Lib | PP Lib |
| MUX-MI | User_Lib | PP Lib |
| DATE | Basic Lib | |
| TIME | Basic Lib | |
| SW | Basic Lib | PP Lib |
| ABS | Basic Lib | |
| REPORT | User_Lib | |
| FILT-1P | User_Lib | PP Lib |
| CON-PU1 | User_Lib Con-Pu1 | PP Lib Con-Pu1(1) |
| PB-R | User_Lib | |
| SR-OA | Basic Lib | |
| MAJ-R | User_Lib | |
| MUXGR-MI | User_Lib | |
| PB-S | User_Lib | |
| XOR | Basic Lib | |
| SQRT | Basic Lib | |
| ADD-MR | Basic Lib | PP Lib |
| ADD-MR1 | Basic Lib | |
| SHIFT/SHIFT-L | Basic Lib | |
| FIFO | | PP Lib |
| DEMUX-MI | User_Lib | PP Lib |
| DER | User_Lib | PP Lib |
| FILT-2P | | PP Lib |
| REG-G | User_Lib | PP Lib |
| FUNG-1V | User_Lib | PP Lib |
| FUNG-2V | User_Lib | User_Lib |
| FUNG-T | User_Lib | PP Lib |
| MAX | Basic Lib | PP Lib |
| MIN | Basic Lib | PP Lib |
| P-1 | User_Lib | User_Lib |
| P-DEADB | User_Lib | User_Lib |
| PI | ControlSimpleLib | PP Lib |
| PDP | User_Lib | User_Lib |
| RAMP | ControlSimpleLib | PP Lib |
| RAMP-S1 | User_Lib | User_Lib |
| TIMER | User_Lib | PP Lib |
| MANSTN | Dummy FB | PP Lib |
| MOTCON | Dummy FB | PP Lib |
| VALVECON | Dummy FB | PP Lib |
| PIDCON | User_Lib | PP Lib |
| PIDCONA | User_Lib | PP Lib |
| GENCON | | PP Lib |
| SEQ | User_Lib, FBD+SFC | PP Lib, FBD+SFC |
| STEP | User_Lib, FBD+SFC | PP Lib, FBD+SFC |
| DIV-MR | Basic Lib | |
| TEXT | User_Lib | |
| RATIOSTN | Dummy FB | PP Lib |
| EXP | Basic Lib | |
| LN | Basic Lib | |
| BGET | Basic Lib | |
| BSET | User_Lib | |
| IIL | User_Lib | |
| ILI | User_Lib | |

| | |
|---|---|
| DRTRA (APC) | User_Lib |
| ANALYSE | User_Lib |
| C2SEL2 | PCITLib |
| CFG2 | PCITLib |
| C2PB1 | PCITLib |
| C2PV1 | PCITLib |
| C3PMV1 | PCITLib |
| C2PV | PCITLib |
| C2PB | PCITLib |
| VIS2PV | User_Lib |
| VIS3MV | User_Lib |
| VIS2PB | User_Lib |
| VISMSTN | User_Lib |
| VIS2SEL | User_Lib |
| VISCFG | User_Lib |

Legend:

PP Lib

- PP Lib solves only a part of all PC elements

Basic Lib

- Remaining part is solved partially by Basic AC800M libraries *or* by UserLib

User_Lib

- **User_Lib contains FBs used as replacement to PC elements without close match in other Libs** already developed, tested and used in many projects

Developed by: Kamil Herman, Slovakia
contact: kamil.herman@herman-automation.com
ABB AC450 and 800xA Specialist Providing a range of Control System design, programming, migrating and commissioning services. We are ready to deliver the complete migrated control software in status „Ready for FAT" or „Ready for Commissioning".