# Context

Given a system that must monitor and control two buildings e.g. North and South each with identical number of bays containing say 50 physical tags each. Depending on the tag, it can further fan out to include derived tags such as set points, timer intervals, controllers, etc.

Many people take accept “data-entry” as past of the job and when a controls engineer gets paid by the hour this is easy money. On the other hand, in a fixed price project or when outsourced competition plays into the mix, a way to automate repetitive tasks becomes a competitive advantage—albeit a small one.

# Current Approach

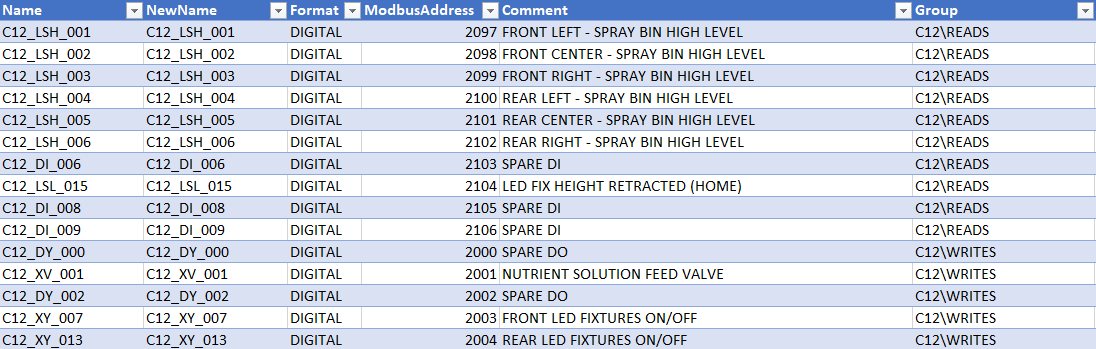
A complementary tool that many uses with the TwinSoft programming environment is the import/export tool. It reads and writes xml files used to import tags into TwinSoft. The files when edited in Excel, remains a time consuming and error prone process. The alternative is to use the CRUD functional and becomes and point and click exercise.

# Helper Tools – tag\_importer.py

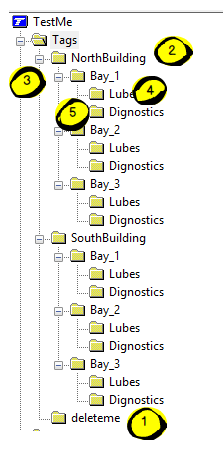
The helper tool started as tag creation tool decoupled from the back-end system. By writing a system specific tag generator one could read generic excel files contain tag information and generate the tags in bulk to import into a system.

There was not a use case at this point given that most of the work was done using [Servelec TG2/LT2](https://www.servelectechnologies.com/servelec-technologies/products-services/remote-telemetry-units/tbox/tbox-lt2/)) RTUs and the tool ended up being TwinSoft centric.

The XML file created from the TwinSoft Import-Export tool contains 23 columns for each tag. An subset of columns is shown below. What is required and how to fill them is outside of the scope of this document and it the help tools satisfy those requirements.



Back to our building scenario. A potential Tag Group structure in TwinSoft is shown below:



The helper tool enforces the rule that all tags must belong to a group and will not proceed with the tag generation otherwise.

## Excel

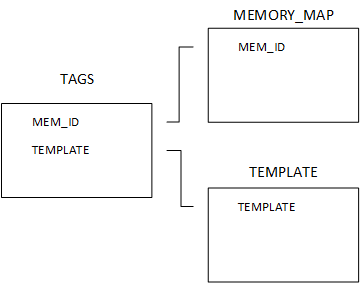
The Excel files requires 3 sheets.

TAGS – contains tag generation and group to memory map

MEMORY\_MAP – specifics Modbus ranges by data type and grouping

TEMPLATE – specifies tag generation rules. e.g. HAND-OFF-AUTO (HOA) tag

The TAG sheet references the other sheets as shown below.



### TEMPLATE SHEET

Template entry

|  |  |
| --- | --- |
| COLUMN | DESCRIPTION |
| TEMPLATE | Template name |
| SUFFIX | Will be added to the tag and most likely as a suffix although not required. Refer to TAGS Sheet description |
| DESCRIPTION | Will be added to the tag description and most likely appended although not required. Refer to TAGS Sheet description |
| TYPE | BOOL, FLOAT, UINT16 (Unsigned 16-bit integer), INT16, UINT32, INT32, UINT8 (TEXT not implemented) |
| INITIAL\_VALUE | Initial value for the tag. -9999 if no initialization required |
| SCRIPT\_VALUE | TwinSoft scripting file containing tags along with script value. . -9999 if no initialization required Not implemented |

### MEMORY\_MAP

|  |  |
| --- | --- |
| COLUMN | DESCRIPTION |
| MEM\_ID | Memory ID representing one more group folders |
| MEM\_TYPE | BOOL, FLOAT, UINT16 (Unsigned 16-bit integer), INT16, UINT32, INT32, UINT8 (TEXT not implemented) |
| START\_ADDRESS | Modbus starting address |
| LENGTH | Quantity required of the given MEM\_TYPE |
| TS\_FORMAT | TwinSoft Format BOOL -> DIGITAL/FALSE, FLOAT -> FLOAT/TRUE, UINT16->16BITS,FALSE, INT16->16BITS,TRUE, UINT32->32BITS,FALSE, INT32->16BITS,TRUE, UINT8->8BITS,FALSE, |
| TS\_SIGNED | Twinsoft Signed TRUE | FALSE |
| COMMENT | Comment not used for anything other that informational |

### TAGS

|  |  |
| --- | --- |
| COLUMN | DESCRIPTION |
| CLASS | BASE for a single Tag Definition, GENERATE for tag generated from a template, MAP group to memory map relationship, one to one or one to many, IGNORE ignore entry |
| TAG\_NAME | TAG\_NAME – name of the tag when class = BASE |
| RULE | FUTURE TODO |
| TAG\_PATTERN | Tag name pattern when class = GENERATE |
| TEMPLATE | Template name used to generate a tag when class = GENERATE |
| TS\_GROUP | TwinSoft group for the given tag |
| MEM\_ID | MEM\_ID for the given TS\_GROUP belongs to |

#### CLASS = GENERATE

The tag pattern typically consists of a leading tag followed by the wildcard character **\***. For example, to generate tags for Hand-Off-Auto (HOA) template one could set TAG\_PATTERN HS\_166\*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CLASS** | **TAG\_PATTERN** | **DESCRIPTION** | **TEMPLATE** | **TS\_GROUP** | **MEM\_ID** |
| GENERATE | **HS\_166\*** | **CHAMBER 1 TEMPERATURE \*** | HOA | CHAMBER 1\SOFTS | CHAMBER 1\SOFTS |

Which states tag HS\_166 seeds a set of HOA template generated tags to be stored under the TwinSoft group CHAMBER 1\SOFTS using CHAMBER 1\SOFT MEMORY\_MAP rules

Given a template entry for HOA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEMPLATE** | **SUFFIX** | **DESCRIPTION** | **TYPE** | **INITIAL\_VALUE** | **SCRIPT\_VALUE** |
| HOA | HOA | HOA STATE (H=1,A=2,O=0) | UINT16 | -9999 | -9999 |
| HOA | \_H | HAND | BOOL | -9999 | -9999 |
| HOA | \_O | OFF | BOOL | -9999 | -9999 |
| HOA | \_A | AUTO | BOOL | -9999 | -9999 |

the generated tags end up as follows:

HS\_166\* leads to HS\_166\_HOA, HS\_166\_H, HS\_166\_O

Note the wildchard character \* gets replaced with the SUFFIX information from the template. One assumes that the [ISA](https://instrumentacionhuertas.files.wordpress.com/2013/07/s_51.pdf) tagging front ends are used.

One could set the TAG\_PATTERN to HS\*\_166 leading to HS\_HOA\_166, HS\_H\_166, HS\_O\_166, HS\_A\_166.

Similarly, for the DESCRIPTION, the wildcard character \* acts as a place holder where the description in the template will be inserted at.

For our HOA example,

CHAMBER 1 TEMPERATURE \* leads to

CHAMBER 1 TEMPERATURE HOA STATE (H=1,A=2,O=0),  
CHAMBER 1 TEMPERATURE HAND,   
CHAMBER 1 TEMPERATURE OFF,   
CHAMBER 1 TEMPERATURE AUTO

How are Modbus addresses calculated? For our HOA example, the template specifies the CHAMBER 1\SOFTS MEMORY\_MAP rules are used to seed the addressing.

The MEMORY\_MAP entry states that for CHAMBER 1\SOFTS BOOL types start at address 1500, FLOATS at 1600, and UINT16 at 1638.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MEM\_ID** | **MEM\_TYPE** | **START\_ADDRESS** | **LENGTH** | **TS\_FORMAT** | **TS\_SIGNED** | **COMMENT** |
| CHAMBER 1 | BOOL | 1030 | 32 | DIGITAL | FALSE | PHYSICAL I/O |
| CHAMBER 1 | FLOAT | 1000 | 14 | FLOAT | TRUE | PHYSICAL I/O |
| **CHAMBER 1\SOFTS** | **BOOL** | **1500** | **60** | **DIGITAL** | **FALSE** | **SOFT I/O** |
| CHAMBER 1\SOFTS | FLOAT | 1600 | 19 | FLOAT | TRUE | SOFT I/O |
| CHAMBER 1\SOFTS | UINT16 | 1638 | 19 | 16BITS | FALSE | SOFT I/O |

One may ask, why so complicated? Back to our building scenario, does one want to create and maintain separate screens for each building and bay or create one set and reference offsets to base Modbus addresses? Larger projects require thinking about Modbus addresses at least when it comes to using TwinSoft.

Back to our HOA example, the template for HOA has 3 BOOL and 1 UINT16. The generate tags and address lead to the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **TAG** | **DESCRIPTION** | **TYPE** | **MODBUS ADDRESS** |
| HS\_166\_HOA | CHAMBER 1 TEMPERATURE HOA STATE (H=1,A=2,O=0) | UINT16 | 1638 |
| HS\_166\_H | CHAMBER 1 TEMPERATURE HAND | BOOL | 1500 |
| HS\_166\_O | CHAMBER 1 TEMPERATURE OFF | BOOL | 1501 |
| HS\_166\_A | CHAMBER 1 TEMPERATURE AUTO | BOOL | 1503 |

Let’s say we also generated HS\_167\* for CHAMBER 1 HUMIDITY \*

|  |  |  |  |
| --- | --- | --- | --- |
| **TAG** | **DESCRIPTION** | **TYPE** | **MODBUS ADDRESS** |
| HS\_166\_HOA | CHAMBER 1 TEMPERATURE HOA STATE (H=1,A=2,O=0) | UINT16 | 1638 |
| HS\_166\_H | CHAMBER 1 TEMPERATURE HAND | BOOL | 1500 |
| HS\_166\_O | CHAMBER 1 TEMPERATURE OFF | BOOL | 1501 |
| HS\_166\_A | CHAMBER 1 TEMPERATURE AUTO | BOOL | 1503 |
| HS\_167\_HOA | CHAMBER 1 HUMIDITY HOA STATE (H=1,A=2,O=0) | UINT16 | 1639 |
| HS\_167\_H | CHAMBER 1 HUMIDITY HAND | BOOL | 1504 |
| HS\_167\_O | CHAMBER 1 HUMIDITY OFF | BOOL | 1505 |
| HS\_167\_A | CHAMBER 1 HUMIDITY AUTO | BOOL | 1506 |

What if we generated 21 tags HS\_166\* to HS\_186\*? Each HOA template has 3 BOOL types which would require 21 tags \* 3 BOOLs = 63 contiguous BOOL addresses. Our MEMORY\_MAP allocated 60 BOOLS starting at 1500. The helper tool would flag this from which we would have to increase our memory map allocation.

When this is done, one can import the generated tags using the Twinsoft Import/Export tool.

Assume that HS\_166\* and HS\_167\* generated tags now reside in TwinSoft and re-exported, and the user generates HS\_168\*? The helper tool resume at the next address available address in the memory map and for this example, BOOL will start at 1507 for HOA.

This is where the CLASS=MAP entry comes into play.

#### CLASS = MAP

Not all groups are represented in tags with CLASS=GENERATE. The mechanism to capture all groups and their corresponding memory maps use entries of CLASS=MAP.

To capture this information, the TAGS sheet requires rows with CLASS=MAP as shown below.

|  |  |  |
| --- | --- | --- |
| **CLASS** | **TS\_GROUP** | **MEM\_ID** |
| MAP | CHAMBER 1 | CHAMBER 1 |
| MAP | CHAMBER 1\TIMER | CHAMBER 1\SOFTS |
| MAP | CHAMBER 1\HOA | CHAMBER 1\SOFTS |

This states, TwinSoft Group CHAMBER 1\TIMER and HOA reference a common CHAMBER 1\SOFTS Memory map. Back to our HOA example, from the CLASS=GENERATE section, should this not be in place, the calculated Modbus address would revert to 1500 rather than the next available one in the TwinSoft export file.

#### CLASS=IGNORE

Any TAGS Sheet entry with Class set to IGNORE will not be processes by tag\_importer.

#### CLASS=BASE

Creating one or more tags is achieved by either the using TwinSoft UI, editing the TwinSoft Export file, or via the following approach. Depending on the context, one may be more convenient than the other.

Creating a tag via the TAGS sheet method, requires entry of CLASS=BASE where a typical entry could be as follows.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CLASS** | **TAG\_NAME** | **TYPE** | **DESCRIPTION** | **TS\_GROUP** | **MEM\_ID** | **ADDRESS** |
| BASE | LT\_001 | FLOAT | GC 1 LEVEL TRANSMITTER | CHAMBER 1 | CHAMBER 1 | 1000 |
|  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| COLUMN | DESCRIPTION |
| TAG\_NAME | Unique tag name for the tag |
| TYPE | BOOL, FLOAT, UINT16 (Unsigned 16-bit integer), INT16, UINT32, INT32, UINT8 (TEXT not implemented) |
| DESCRIPTION | Tag description |
| TS\_GROUP | TwinSoft group for the given tag |
| MEM\_ID | MEM\_ID for the given TS\_GROUP belongs to |
| ADDRESS | If left blank, will use the next available address for the give type as per MEMORY\_MAP. If given a value, will be assigned this address, if it still falls in the MEMORY\_MAP. |
| INITIAL\_VALUE | Initial value for the tag. -9999 if no initialization required |

## How to Start.

1. Create a placeholder folder e.g. deleteme and a tag inside it of any type with a modbus address that is not in the targeting range.
2. Using a copy of the tags\_degs.xlsx, start mapping out the memory map for your solution
3. Modify the TEMPLATE tab as required for your given solution
4. Define all TS\_GROUP/MEM\_ID for CLASS=BASE in the TAGS sheet
5. Create GENERATE tags as required
6. Create BASE tags as required

Typically, an I/O list exists as part of the engineering process. The TABS, MEMORY\_MAP, TEMPLATE sheets could exist in I/O list spreadsheet. If the required sheet name and columns exists, the helper tool can read and process the required information, otherwise a corresponding error message provides feedback.

# Requirements to Run

Python 3.7.5 +

Pandas

import pandas as pd

#import argparse

#import sys

import logging

import click

# Examples