

ДОКУМЕНТАЦИЯ
На программу
«ИТР
Конфигуратор»

16.05.23

Оглавление

Вкладка Regul	3
Преобразование xml файлов ПО	3
Работа с файлом ЭЛСИ-ТМК.xml (крейт)	4
Работа с файлом _Application.xml (приложение)	7
Этап 1. Текстовые замены	8
Этап 2. Модификация dlnitFlag	9
Этап 3. Создание триггеров	9
Этап 4. Преобразование глобальных Retain переменных в Persistent	10
Этап 5. Модификация секции FB	11
Этап 6. Резервирование	11
Этап 7. Создание MBS_GVL для modbus каналов	16
Генерация Modbus устройств из csv	19
Генерация Modbus устройств по .xlsx таблице	23
Вкладка MKLogic	25
Создание .csv файла привязки переменных	25
Модификация POU	25
Модификация индексов IEC	26
Замена циклов FOR на WHILE	27
Модификация ФБ	28
Вкладка Alpha	30
Генерация карт адресов для DevStudio из csv	30
Генерация Modbus карты из карт уставок	30
Alpha.HMI	31
Пропатчить на водяной знак	31
Сгенерить функцию updateAlarm	33
Получить таблицу функций	34
Генератор объектов	35
Вкладка Инф. обесп.	37
Вкладка Другое	41
ЭЛСИ-ТМК	41
Проставить Deny Request флаги	41

Вкладка Regul

Преобразование xml файлов ПО

ИТР Конфигуратор

Настройки Справка

Regul MKLogic Alpha Инф. обесп. Другое

Преобразование xml файлов ПО

Версия устройств

☒ RegulBus OS

☒ Iec104Slave OS

Выбор ПЛК

Regul R500

Загрузить xml

Генерация Modbus устройств из csv

Доступно шаблонов: 0

Путь

Версия устройств 1.6.5.1

☐ Преобразовать шаблоны к OS версии

Загрузить csv

Генерация Modbus устройств по .xlsx таблице (name, ip, template)

Открыть таблицу .xlsx

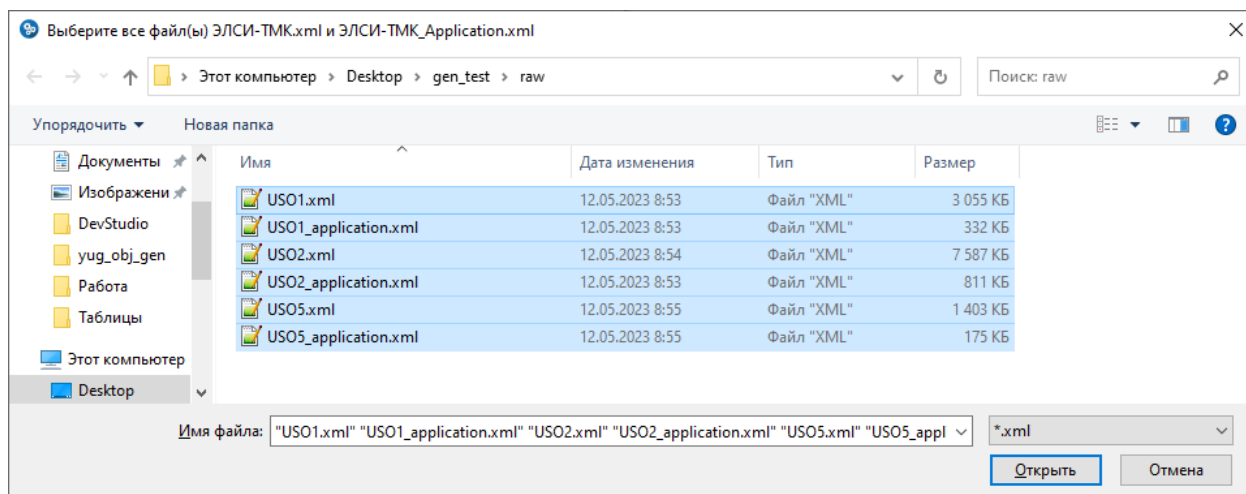
Рисунок

Нажатие кнопки «Загрузить XML» открывает диалоговое окно для выбора файлов ЭЛСИ-ТМК.xml и ЭЛСИ-ТМК_Application.xml, которые предварительно были созданы программой ACS Configurator.

Одновременно можно выбрать несколько файлов с разных УСО. Программа автоматически попытается загрузить рядом лежащий файл ЭЛСИ-ТМК.xml, если был выбран только одноименный _Application.xml и наоборот.

При выборе файла _Application.xml и отсутствии рядом файла ЭЛСИ-ТМК.xml, программа будет модифицировать только приложение контроллера.

Если же выбрать только ЭЛСИ-ТМК.xml, без файла _Application рядом, то программа сделает ровным счетом ничего.



Рисунок

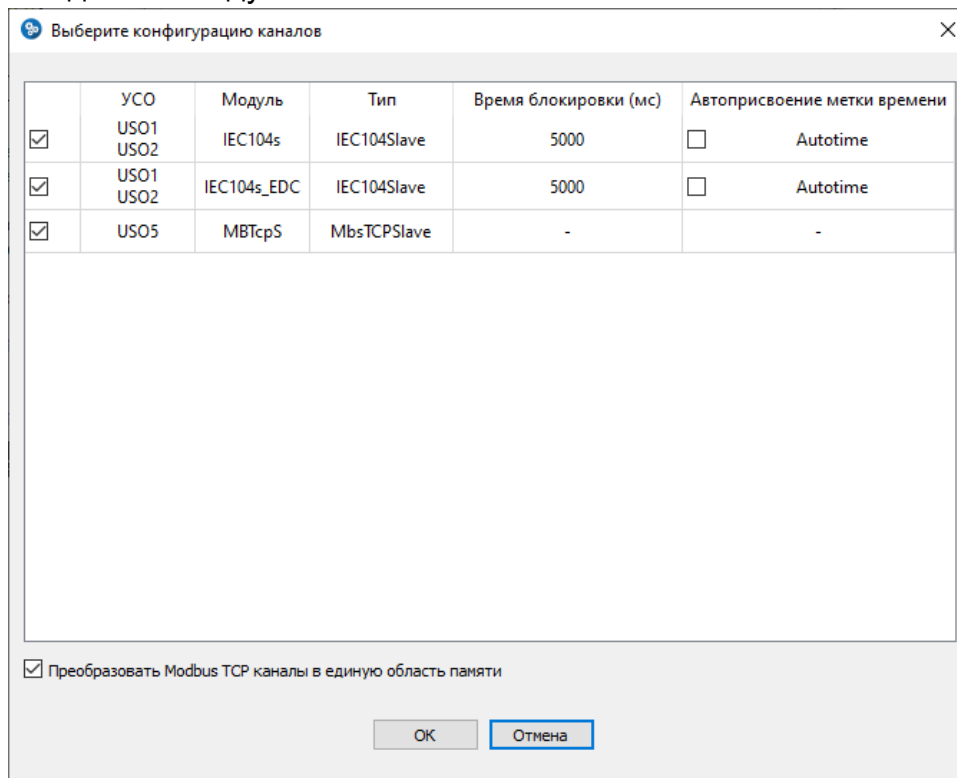
Дальнейшее преобразование этих двух типов файлов делится на два этапа:

- Работа с файлом ЭЛСИ-ТМК.xml (крейт)
- Работа с файлом _Application.xml (приложение)

Работа с файлом ЭЛСИ-ТМК.xml (крейт)

Конечная цель работы с крейтом – создание .xml файлов с Модбас или МЭК каналами (адресами и привязанными переменными), которые затем можно вручную или в автоматическом режиме импортировать в Epsilon LD/Astra IDE.

Просматривая каждый крейт, программа попытается найти все slave модули типа IEC104s или MBTcрS внутри него. В случае успеха откроется следующее окно со списком найденных модулей:

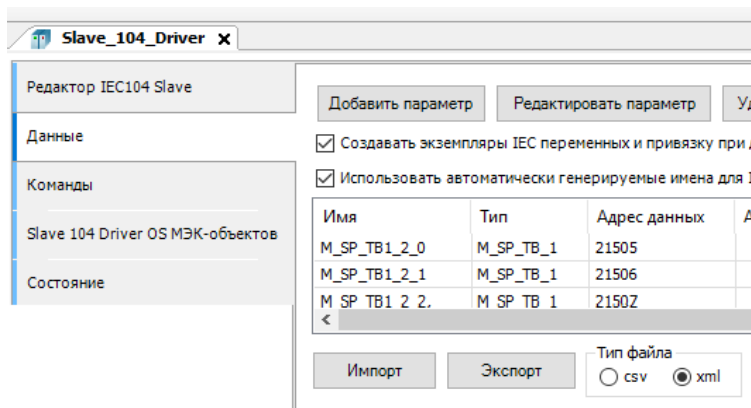


Рисунок

Для каждого выбранного МЭК модуля в УСО создается два файла:

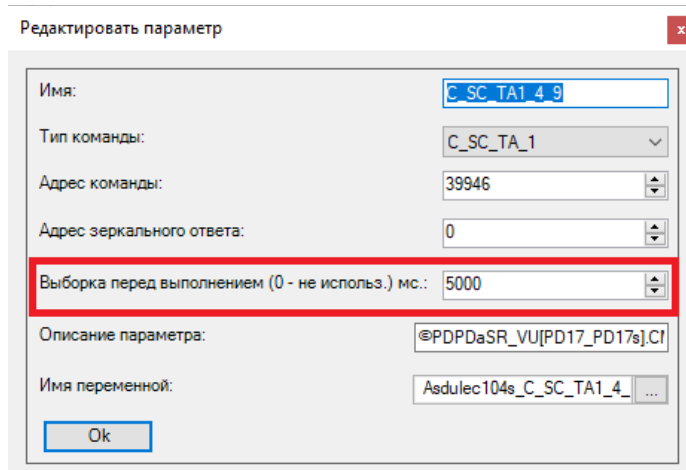
- REGUL_IEC104_ST_CMD.iec104cmd.xml – для каналов команд
- REGUL_IEC104_ST_DATA.iec104data.xml – для каналов данных

Которые можно загрузить с помощью кнопки Импорт в соответствующую вкладку Данные или Команды среды Epsilon LD/Astra IDE в модуль Slave_104_Driver:



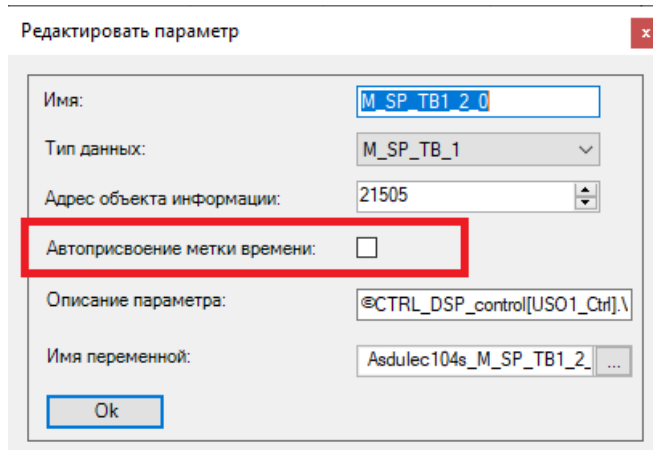
Рисунок

Настройка «Время блокировки (мс)» применима к каналам команд:



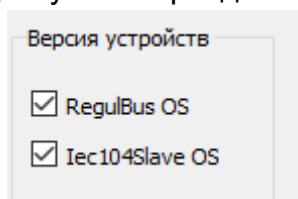
Рисунок

А опция «Автоприсвоение метки времени» к каналам данных:



Рисунок

Опция Iec104Slave OS, доступная в разделе Версия устройств:



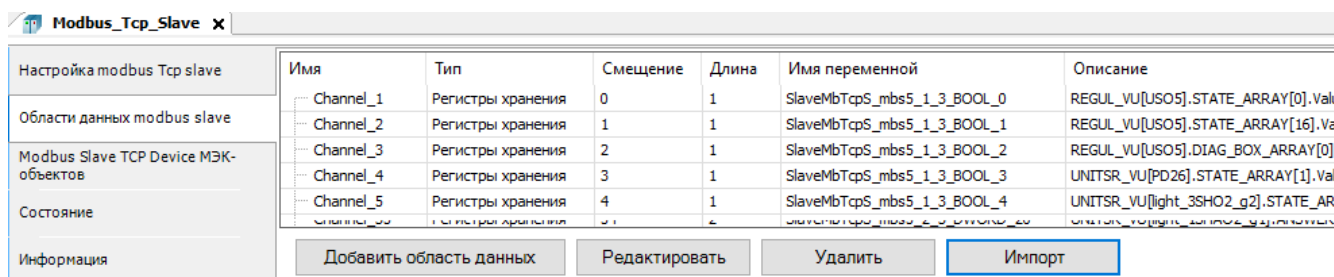
Рисунок

Влияет только на формат времени блокировки в каналах команд. Iec104Slave OS – отмечено, значит время попадет в мс (5000), иначе в сек (5).

Для каждого выбранного Модбас модуля создается один файл:

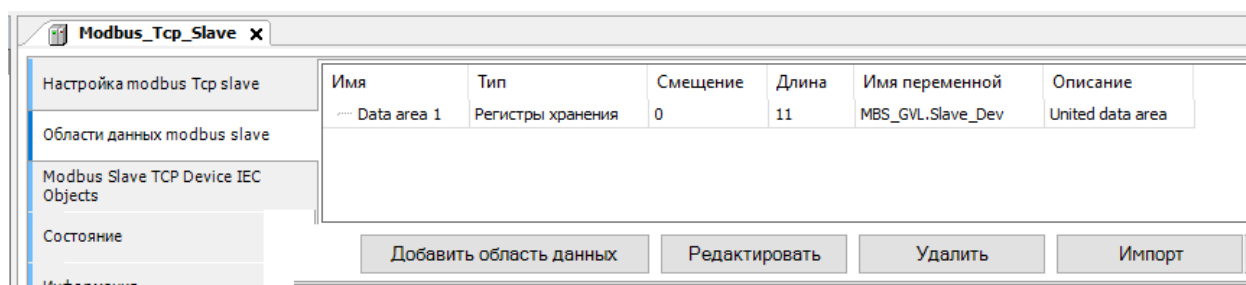
- REGUL_XML.mb_direct_channels.xml

Если опция «Преобразовать Modbus TCP каналы в единую область памяти» не была отмечена, файл REGUL_XML.mb_direct_channels.xml будет содержать все имеющиеся каналы, вместе с привязанной переменной, описанием, адресом и др. При импорте в модуль Modbus_Tcp_Slave, каналы будут выглядеть следующим образом:



Рисунок

Если же опция «Преобразовать Modbus TCP каналы в единую область памяти» была отмечена, то создается файл REGUL_XML.mb_direct_channels.xml всего с одним каналом «Data area 1», который является массивом достаточной длины для размещения всех каналов в памяти друг за другом без пробелов. Этот канал имеет маппинг на переменную MBS_GVL.Slave_Dev.



Рисунок

Так же программа создаст в _Application.xml файл MBS_GVL, где сопоставит каждой modbus переменной ее адрес в массиве Slave_Dev с помощью операции ADR.

Было выяснено, что при большом количестве каналов (сотни, тысячи), создание единой области памяти существенно влияет на производительность ПЛК, уменьшая загрузку ЦП.

```

1 {attribute 'subsequent' := ''}
2 {attribute 'pack_mode' := '1'}
3 VAR_GLOBAL
4   Slave_Dev: ARRAY[0..10] OF WORD;
5   ///DIAG_SHA_VU[DIAG_28_1_SHA_1].State_Box.0
6   Slave_mbs5_1_3_BOOL_0: POINTER TO WORD := (ADR(Slave_Dev) + (0* SIZEOF(WORD)));
7   ///DIAG_SHA_VU[DIAG_28_1_SHA_1].State_PLC.0
8   Slave_mbs5_1_3_BOOL_1: POINTER TO WORD := (ADR(Slave_Dev) + (1* SIZEOF(WORD)));
9   ///DIAG_SHA_VU[DIAG_28_1_SHA_1].ModuleFLT.0
10  Slave_mbs5_1_3_BOOL_2: POINTER TO WORD := (ADR(Slave_Dev) + (2* SIZEOF(WORD)));
11  ///TS_OUT_VU[SHU1_LHH]
12  Slave_mbs5_1_3_BOOL_3: POINTER TO WORD := (ADR(Slave_Dev) + (3* SIZEOF(WORD)));
13  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].Cnt
14  Slave_mbs5_2_3_UINT_0: POINTER TO UINT := (ADR(Slave_Dev) + (10* SIZEOF(WORD)));
15  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Year
16  Slave_mbs5_3_3_WORD_0: POINTER TO WORD := (ADR(Slave_Dev) + (4* SIZEOF(WORD)));
17  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Month
18  Slave_mbs5_3_3_WORD_1: POINTER TO WORD := (ADR(Slave_Dev) + (5* SIZEOF(WORD)));
19  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Day
20  Slave_mbs5_3_3_WORD_2: POINTER TO WORD := (ADR(Slave_Dev) + (6* SIZEOF(WORD)));
21  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Hour
22  Slave_mbs5_3_3_WORD_3: POINTER TO WORD := (ADR(Slave_Dev) + (7* SIZEOF(WORD)));
23  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Minute
24  Slave_mbs5_3_3_WORD_4: POINTER TO WORD := (ADR(Slave_Dev) + (8* SIZEOF(WORD)));
25  ///DIAG_SHA_VU[DIAG_28_1_SHA_1].CurTimeDate.MB_Sec
26  Slave_mbs5_3_3_WORD_5: POINTER TO WORD := (ADR(Slave_Dev) + (9* SIZEOF(WORD)));
27 END_VAR

```

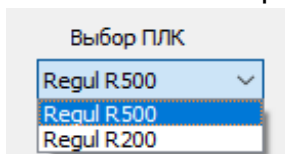
Рисунок

Работа с файлом _Application.xml (приложение)

Конечная цель работы с файлом _Application.xml – создание файла REGUL_Application.xml, который просто является модифицированной версией приложения для работы с ПЛК Регул.

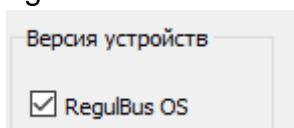
Эту версию приложения, конечно, можно импортировать и вручную в среду Epsilon LD/Astra IDE, но настоятельно рекомендуется производить импорт с помощью python скриптов CodesysLoader.py/CodesysLoader_Astra.py, так как скрипты вносят финальные правки, необходимые для правильной работы программы (да и просто для компиляции без ошибок).

На модификацию приложения влияет как выбранный ПЛК:



Рисунок

Так и отмеченный чекбокс RegulBus OS:



Рисунок

Модификацию приложения можно разбить на следующие этапы:

- Текстовые замены
- Модификация dlnitFlag
- Создание триггеров
- Преобразование глобальных Retain переменных в Persistent
- Модификация секции FB
- Резервирование
- Создание MBS_GVL для modbus каналов

Этап 1. Текстовые замены

Построчно читаем файл приложения и модифицируем строку при необходимости.

1. Удаление строчек .control

Например, строки:

A13_R500_DO_32.control := 2

Если строка содержит

.control :=

то она удаляется.

2. Замены

Ha	.Value__
Ha	:= _imit_;

3. В зависимости от выбранных настроек, меняем обращения к модулям DI/DO

Если выбран **RegulBus OS** И ПЛК = **Regul R500**

Замены

Ha	_R500_DI_32.value
Ha	_Discrets
Ha	_R500_DO_32.value
Ha	_Discrets

Иначе, если ПЛК = **Regul R200**

Замены

Ha	_R500_DI_32.value
Ha	.Inputs_v2^.Discrets
Ha	_R500_DO_32.value
Ha	.Outputs_v2^.Discrets

Иначе, если ПКЛ = **Regul R500**

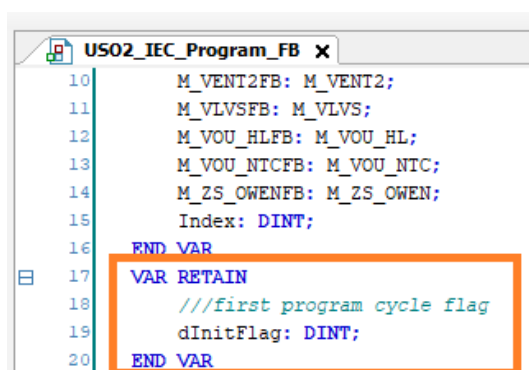
Замены

На	_R500_DI_32.value
	.Inputs_v1^.Discrets
На	_R500_DO_32.value
	_Discrets

Этап 2. Модификация dInitFlag

В каждом POU происходит поиск переменной dInitFlag и снимается флаг retain с true на false. Что приводит к следующим изменениям:

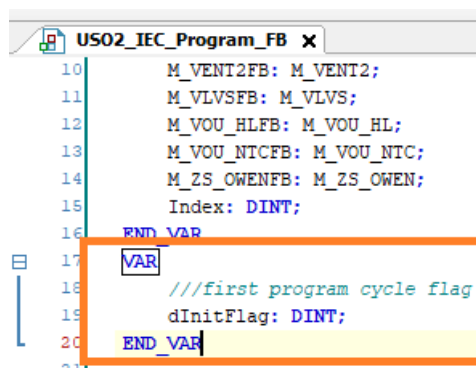
Было:



```
10 M_VENT2FB: M_VENT2;  
11 M_VLVSFB: M_VLVS;  
12 M_VOU_HLFB: M_VOU_HL;  
13 M_VOU_NTFCB: M_VOU_NTC;  
14 M_ZS_OWENFB: M_ZS_OWEN;  
15 Index: DINT;  
16 END VAR  
17 VAR RETAIN  
18   ///first program cycle flag  
19   dInitFlag: DINT;  
20 END VAR
```

Рисунок

Стало:



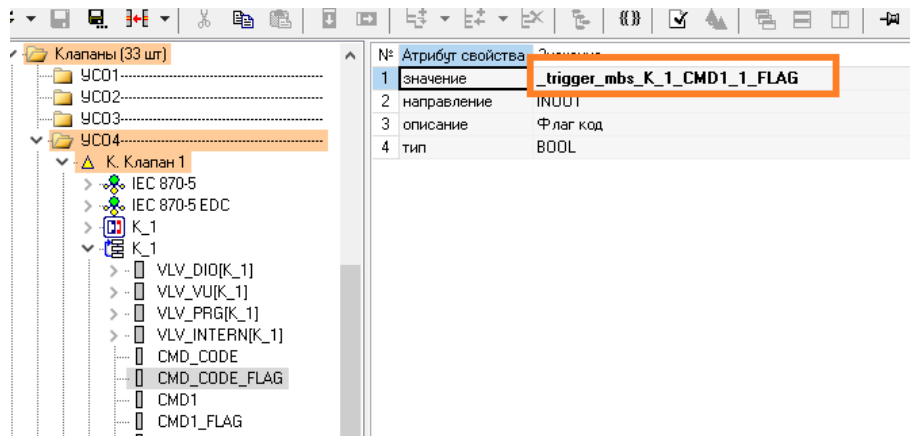
```
10 M_VENT2FB: M_VENT2;  
11 M_VLVSFB: M_VLVS;  
12 M_VOU_HLFB: M_VOU_HL;  
13 M_VOU_NTFCB: M_VOU_NTC;  
14 M_ZS_OWENFB: M_ZS_OWEN;  
15 Index: DINT;  
16 END VAR  
17 VAR  
18   ///first program cycle flag  
19   dInitFlag: DINT;  
20 END VAR
```

Рисунок

Этап 3. Создание триггеров

Триггеры необходимы для правильной работы команд в Modbus Serial Outer Slave устройствах. Эти устройства конфигурируются так, что каналы команд имеют тип канала – Триггер. И имеют соответствующий маппинг на глобальные переменные _trigger_mbs_XX_CMD_FLAG. Когда ФБ устройства посылает команду на нижний уровень, он дергает этот флаг, тем самым синхронизируя посылку команды в Modbus Serial Outer Slave.

Переменные не создаются автоматически, они просто начинают использоваться при работе в программе ACS Configurator:



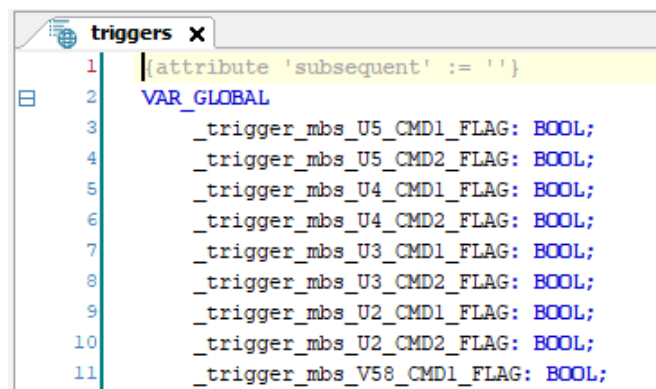
Рисунок

Соответственно конвертер проходится по каждой строке в _Application.xml, и, если видит строку содержащую

trigger

То он складывает полное имя переменной триггера в массив.

Затем создает глобальный объект triggers, содержащий все уникальные переменные _trigger типа BOOL:

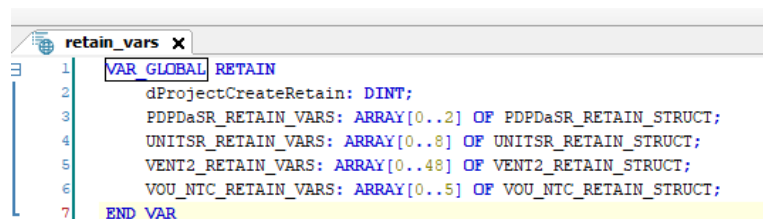


Рисунок

Этап 4. Преобразование глобальных Retain переменных в Persistent

Программа проходится по всем глобальным переменным и если находит глобальные переменные с именем retain_vars, то добавляет им атрибут persistent со значением true и меняет имя с retain_vars на PersistentVars.

Было:



Рисунок

Стало:

```

1  VAR GLOBAL RETAIN PERSISTENT
2
3  dProjectCreateRetain: DINT;
4  PDPDaSR_RETAIN_VARS: ARRAY[0..2] OF PDPDaSR_RETAIN_STRUCT;
5  UNITSR_RETAIN_VARS: ARRAY[0..8] OF UNITSR_RETAIN_STRUCT;
6  VENT2_RETAIN_VARS: ARRAY[0..48] OF VENT2_RETAIN_STRUCT;
7  VOU_NTC_RETAIN_VARS: ARRAY[0..5] OF VOU_NTC_RETAIN_STRUCT;
8
9  END_VAR

```

Рисунок

Этап 5. Модификация секции FB

Вырезаем из секций FB все строки под заголовком (* variables setting *) и вставляем их в конец field секции. Не помню, зачем точно это нужно, вроде для правильной работы блоков имитации.

```

1  PROGRAM USO2_IEC_Program_FB
2  VAR
3      M_ACRONFB: M_ACRON;
4      M_COMPFB: M_COMP;
5      M_CONTROLFB: M_CONTROL;
6      M_PDPDaSRFB: M_PDPDaSR;
7      M_REGULRTFB: M_REGULRT;
8      M_SSFB: M_SS;
9      M_UNITSRFB: M_UNITSR;
10     M_VENT2FB: M_VENT2;
11     M_VLVsFB: M_VLVs;
12     M_VOU_HLFB: M_VOU_HL;
13     M_VOU_NTCFB: M_VOU_NTC;
14
15     ZS_OWEN_PRG[ZS_2_1].LINK_PRE := 3000;
16     dInitFlag := 0512230853;
17 END_IF;
18
19 (* variables setting *)
20 ACRON_DIO[Akron_1].Value_LW := mbs_Akron_1_CH1_Value_LW;
21 ACRON_DIO[Akron_1].Value_HW := mbs_Akron_1_CH1_Value_HW;
22 ACRON_DIO[Akron_1].Moto_LW := mbs_Akron_1_CH1_Moto_LW;
23 ACRON_DIO[Akron_1].Moto_HW := mbs_Akron_1_CH1_Moto_HW;
24 ACRON_DIO[Akron_1].FT_LW := mbs_Akron_1_CH1_FT_LW;
25 ACRON_DIO[Akron_1].FT_HW := mbs_Akron_1_CH1_FT_HW;
26 ACRON_DIO[Akron_1].Rate_LW := mbs_Akron_1_CH1_Rate_LW;
27 ACRON_DIO[Akron_1].Rate_HW := mbs_Akron_1_CH1_Rate_HW;
28 ACRON_DIO[Akron_1].NoLink := NOT mbs_Akron_1_Link;
29 COMP_DI[Module_A4].DISCRETS := A4_Discrets;
30 COMP_DI[Module_A5].DISCRETS := A5_Discrets;

```

```

1  PROGRAM USO2_IEC_Program_field
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Рисунок

Этап 6. Резервирование

Для начала необходимо понять тип резервирования. Программа проходит по всем POU и ищет переменную, которая содержит слово

M_REGUL

```

1  PROGRAM USO2_IEC_Program_FB
2  VAR
3      M_ACRONFB: M_ACRON;
4      M_COMPFB: M_COMP;
5      M_CONTROLFB: M_CONTROL;
6      M_PDPDaSRFB: M_PDPDaSR;
7      M_REGULRTFB: M_REGULRT;
8      M_SSFB: M_SS;
9      M_UNITSRFB: M_UNITSR;
10     M_VENT2FB: M_VENT2;
11     M_VLVsFB: M_VLVs;

```

Рисунок

Затем она смотрит на название типа этой переменной, если тип называется **M_REGUL**, то тип резервирования = 0 (обычное).

Если **M_REGULOS**, то тип резервирования = 1 (OC).

Если **M_REGULRT**, то тип резервирования = 2 (PT).

Иначе тип резервирования = 3 (без резервирования – нет модификаций).

Далее исходя из типа резервирования происходит модификация.

1. Тип резервирования 0 (обычное). Тип блока диагностики УСО - **M_REGUL**

На данный момент данный тип резервирования можно увидеть на станции Терехово ЭМС.

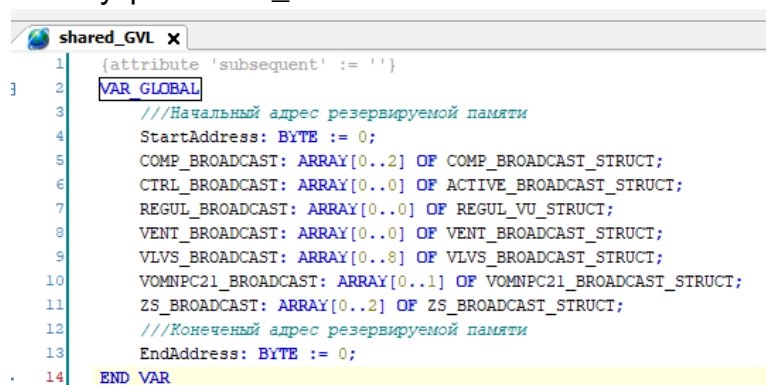
Создается файл `shared_GVL` с атрибутом **subsequent**, который позволяет расположить все объявленные внутри переменные в памяти друг за другом без **выравнивания**.

Так же создаются переменные `StartAddress` и `EndAddress`, задающие границы области резервирования между контроллерами.

Затем проходимся по всем глобальным переменным и если их имя оканчивается на

Или	<code>_BROADCAST</code>
	<code>_SHARED</code>

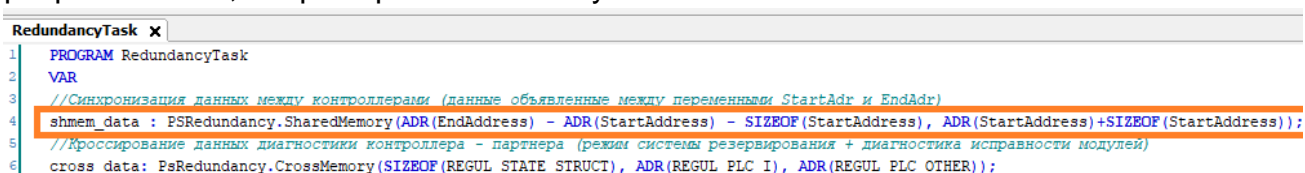
То они вырезаются из своего текущего месторасположения и вставляются внутрь `shared_GVL`:



```
1 {attribute 'subsequent' := ''}  
2 VAR GLOBAL  
3  
4   ///Начальный адрес резервируемой памяти  
5   StartAddress: BYTE := 0;  
6   COMP_BROADCAST: ARRAY[0..2] OF COMP_BROADCAST_STRUCT;  
7   CTRL_BROADCAST: ARRAY[0..0] OF ACTIVE_BROADCAST_STRUCT;  
8   REGUL_BROADCAST: ARRAY[0..0] OF REGUL_VU_STRUCT;  
9   VENT_BROADCAST: ARRAY[0..0] OF VENT_BROADCAST_STRUCT;  
10  VLVS_BROADCAST: ARRAY[0..8] OF VLVS_BROADCAST_STRUCT;  
11  VOMNPC21_BROADCAST: ARRAY[0..1] OF VOMNPC21_BROADCAST_STRUCT;  
12  ZS_BROADCAST: ARRAY[0..2] OF ZS_BROADCAST_STRUCT;  
13  ///Конечный адрес резервируемой памяти  
14  EndAddress: BYTE := 0;  
END_VAR
```

Рисунок

Затем область памяти между `StartAddress` и `EndAddress` уже используется разработчиком, например в `RedundancyTask`:



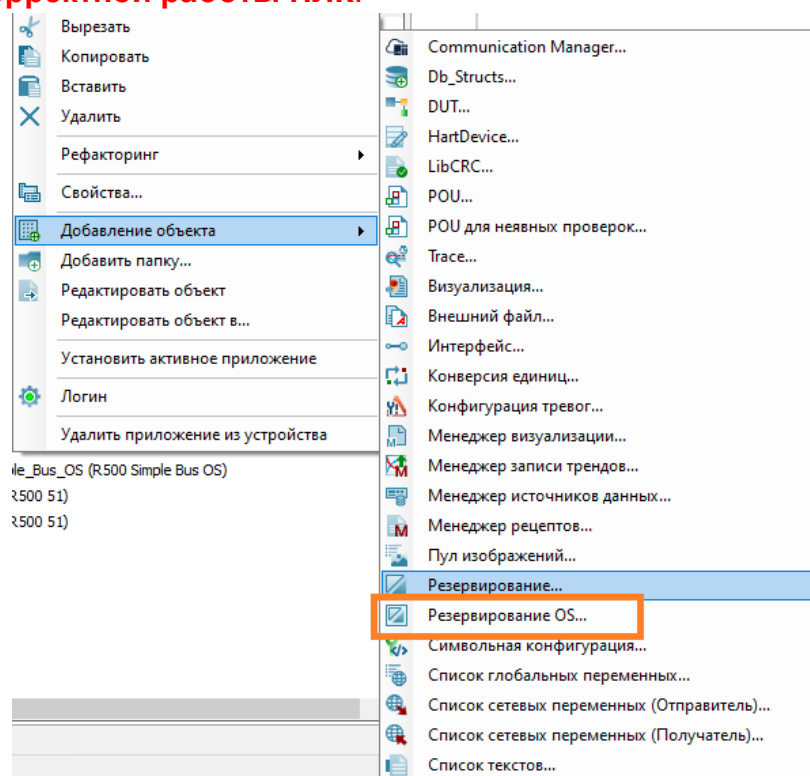
```
1 PROGRAM RedundancyTask  
2 VAR  
3   ///Синхронизация данных между контроллерами (данные объявленные между переменными StartAdr и EndAdr)  
4   shmem_data : PSRedundancy.SharedMemory(ADR(EndAddress) - ADR(StartAddress) - SIZEOF(StartAddress), ADR(StartAddress)+SIZEOF(StartAddress));  
5   ///Кроссирование данных диагностики контроллера - партнера (режим системы резервирования + диагностика исправности модулей)  
6   cross_data: PsRedundancy.CrossMemory(SIZEOF(REGUL_STATE_STRUCT), ADR(REGUL_PLC_I), ADR(REGUL_PLC_OTHER));
```

Рисунок

SharedMemory - При синхронизации данные копируются от ведущего модуля ЦП к ведомому, заменяя предыдущие данные в ведомом ЦП.

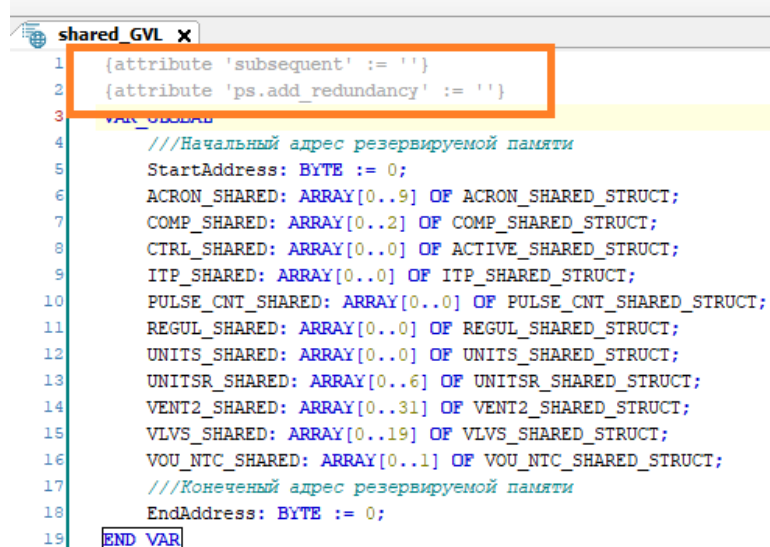
2. Тип резервирования 1 (OS). Тип блока диагностики YCO - **M_REGULOS**
Вместо обычного объекта Redundancy в дереве проекта для этого типа резервирования используется RedundancyOS.

Пока данный тип резервирования не используется в проектах из-за некорректной работы ПЛК.



Рисунок

Так же, как и с типом резервирования 0, создается файл shared_GVL, но помимо атрибута **subsequent**, добавляется так же атрибут **ps.add_redundancy**.



Рисунок

Из документации [Redundancy User Guide DPA 302 4 v2 6 rus.pdf](#):

Можно указать переменные резервирования в коде проекта, для этого применяется атрибут «ps.add_redundancy». Переменные с таким атрибутом будут автоматически отмечены в таблице осветлённым флажком без возможности редактирования (Рисунок 48). При этом сохраняются все правила относительно недопустимых объектов для синхронизации.

Рисунок

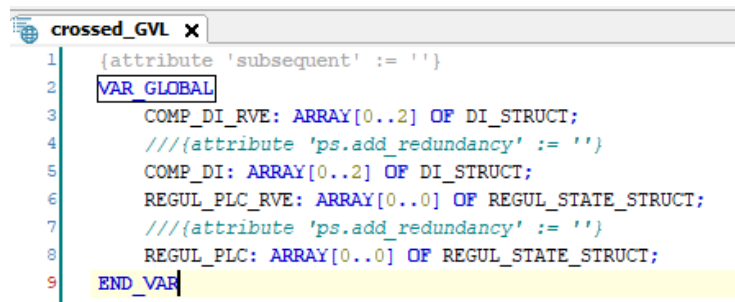
В shared_GVL попадают переменные, которые оканчиваются на

_SHARED

В отличие от предыдущего типа резервирования, переменные оканчивающиеся на _BROADCAST не трогаем.

Так же создается документ crossed_gvl с атрибутом **subsequent**.

CrossMemory - при синхронизации осуществляет обмен данными между модулями ЦП.



```
1 {attribute 'subsequent' := ''}
2 VAR GLOBAL
3   COMP_DI_RVE: ARRAY[0..2] OF DI_STRUCT;
4   ///{attribute 'ps.add_redundancy' := ''}
5   COMP_DI: ARRAY[0..2] OF DI_STRUCT;
6   REGUL_PLC_RVE: ARRAY[0..0] OF REGUL_STATE_STRUCT;
7   ///{attribute 'ps.add_redundancy' := ''}
8   REGUL_PLC: ARRAY[0..0] OF REGUL_STATE_STRUCT;
9 END_VAR
```

Рисунок

В crossed_gvl попадают переменные, которые оканчиваются на

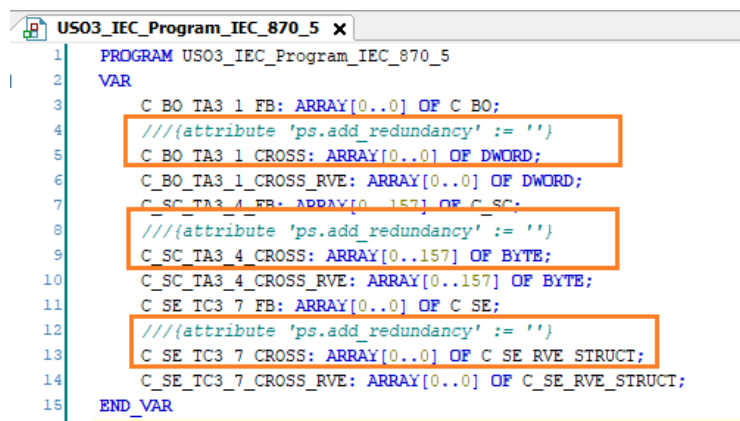
_RVE

А так же одноименные переменные, без этой приставки. Например name123 и name123_RVE.

Одноименным переменным добавляется атрибут **ps.add_redundancy**.

Последним этапом будет добавление атрибута **ps.add_redundancy** переменным внутри любых POU, которые имеют окончание

_CROSS



```
1 PROGRAM US03_IEC_Program_IEC_870_5
2 VAR
3   C_BO_TA3_1_FB: ARRAY[0..0] OF C_BO;
4   ///{attribute 'ps.add_redundancy' := ''}
5   C_BO_TA3_1_CROSS: ARRAY[0..0] OF DWORD;
6   C_BO_TA3_1_CROSS_RVE: ARRAY[0..0] OF DWORD;
7   C_SC_TA3_4_FB: ARRAY[0..157] OF C_SC;
8   ///{attribute 'ps.add_redundancy' := ''}
9   C_SC_TA3_4_CROSS: ARRAY[0..157] OF BYTE;
10  C_SC_TA3_4_CROSS_RVE: ARRAY[0..157] OF BYTE;
11  C_SE_TC3_7_FB: ARRAY[0..0] OF C_SE;
12  ///{attribute 'ps.add_redundancy' := ''}
13  C_SE_TC3_7_CROSS: ARRAY[0..0] OF C_SE_RVE_STRUCT;
14  C_SE_TC3_7_CROSS_RVE: ARRAY[0..0] OF C_SE_RVE_STRUCT;
15 END_VAR
```

Рисунок

Сейчас программа добавляет закоментированные атрибуты, их нужно вручную раскомментировать (баг Astra IDE).

3. Тип резервирования 3 (RT). Тип блока диагностики УСО - **M_REGULRT**

Так же, как и в типе резервирования OS мы добавляем файлы shared_GVL и crossed_GVL. Но есть отличия.

shared_GVL и crossed_GVL создаются только с атрибутом **subsequent**. Атрибут **ps.add_redundancy** больше не используется.

В конце shared_GVL добавляется строка с объявлением переменной shared_data.

```
shared_data: PSRedundancy.SharedMemory(((ADR(EndAddress) - ADR(StartAddress)) -
SIZEOF(StartAddress)), (ADR(StartAddress) + SIZEOF(StartAddress)));
```

```
shared_GVL x
1 {attribute 'subsequent' := ''}
2 VAR GLOBAL
3   ///Начальный адрес резервируемой памяти
4   StartAddress: BYTE := 0;
5   COMP_SHARED: ARRAY[0..2] OF COMP_SHARED_STRUCT;
6   CTRL_SHARED: ARRAY[0..0] OF ACTIVE_SHARED_STRUCT;
7   ITP_SHARED: ARRAY[0..0] OF ITP_SHARED_STRUCT;
8   PDPDaSR_SHARED: ARRAY[0..1] OF PDPDaSR_SHARED_STRUCT;
9   REGUL_SHARED: ARRAY[0..0] OF REGUL_SHARED_STRUCT;
10  SS_SHARED: ARRAY[0..2] OF SS_SHARED_STRUCT;
11  UNIT_SHARED: ARRAY[0..0] OF UNIT_SHARED_STRUCT;
12  UNITSR_SHARED: ARRAY[0..7] OF UNITSR_SHARED_STRUCT;
13  ///Конечный адрес резервируемой памяти
14  EndAddress: BYTE := 0;
15  shared_data: PSRedundancy.SharedMemory(((ADR(EndAddress) - ADR(StartAddress)) - SIZEOF(StartAddress)), (ADR(StartAddress) + SIZEOF(StartAddress)));
16 END_VAR
```

Рисунок

В конце crossed_GVL добавляются строки с объявлением переменных cross_data_x. Где x – порядковый номер пары переменных name_RVE и name.

```
crossed_GVL x
1 {attribute 'subsequent' := ''}
2 VAR GLOBAL
3   COMP_DI_RVE: ARRAY[0..2] OF DI_STRUCT;
4   COMP_DI: ARRAY[0..2] OF DI_STRUCT;
5   REGUL_PLC_RVE: ARRAY[0..0] OF REGUL_STATE_STRUCT;
6   REGUL_PLC: ARRAY[0..0] OF REGUL_STATE_STRUCT;
7   cross_data_1: PSRedundancy.CrossMemory(SIZEOF(COMP_DI), ADR(COMP_DI), ADR(COMP_DI_RVE));
8   cross_data_2: PSRedundancy.CrossMemory(SIZEOF(REGUL_PLC), ADR(REGUL_PLC), ADR(REGUL_PLC_RVE));
9 END_VAR
```

Рисунок

Последним этапом будет добавление переменных cross_data_X внутри любых POU, где объявлены пары переменных типа

И

_CROSS

_CROSS_RVE


```

USO4_IEC_Program_IEC_870_5
1 PROGRAM USO4_IEC_Program_IEC_870_5
2 VAR
3   C_BO_TA4_1_FB: ARRAY[0..0] OF C_BO;
4   C_BO_TA4_1_CROSS: ARRAY[0..0] OF DWORD;
5   C_BO_TA4_1_CROSS_RVE: ARRAY[0..0] OF DWORD;
6   C_SC_TA4_4_FB: ARRAY[0..355] OF C_SC;
7   C_SC_TA4_4_CROSS: ARRAY[0..355] OF BYTE;
8   C_SC_TA4_4_CROSS_RVE: ARRAY[0..355] OF BYTE;
9   C_SE_TC4_8_FB: ARRAY[0..3] OF C_SE;
10  C_SE_TC4_8_CROSS: ARRAY[0..3] OF C_SE_RVE_STRUCT;
11  C_SE_TC4_8_CROSS_RVE: ARRAY[0..3] OF C_SE_RVE_STRUCT;
12  cross_data_1: PsRedundancy.CrossMemory(SIZEOF(C_BO_TA4_1_CROSS), ADR(C_BO_TA4_1_CROSS), ADR(C_BO_TA4_1_CROSS_RVE));
13  cross_data_2: PsRedundancy.CrossMemory(SIZEOF(C_SC_TA4_4_CROSS), ADR(C_SC_TA4_4_CROSS), ADR(C_SC_TA4_4_CROSS_RVE));
14  cross_data_3: PsRedundancy.CrossMemory(SIZEOF(C_SE_TC4_8_CROSS), ADR(C_SE_TC4_8_CROSS), ADR(C_SE_TC4_8_CROSS_RVE));
15 END_VAR

```

Рисунок

Этап 7. Создание MBS_GVL для modbus каналов

Это опциональный этап, делается лишь в случае наличия модуля MbsTcpSlave с каналами.

Если на этапе выбора конфигурации каналов не была отмечена опция «Преобразовать Modbus TCP каналы в единую область памяти»

Выберите конфигурацию каналов

	УСО	Модуль	Тип	Время блокировки (мс)	Автоприсвоение метки времени
<input checked="" type="checkbox"/>	USO5	MBTcpS	MbsTCPSlave	-	-

☐ Преобразовать Modbus TCP каналы в единую область памяти

OK Отмена

Рисунок

То создается файл глобальных переменных MBS_GVL с атрибутом **subsequent** и просто со списком переменных SlaveMbTcpS_mbs_xxx:

```

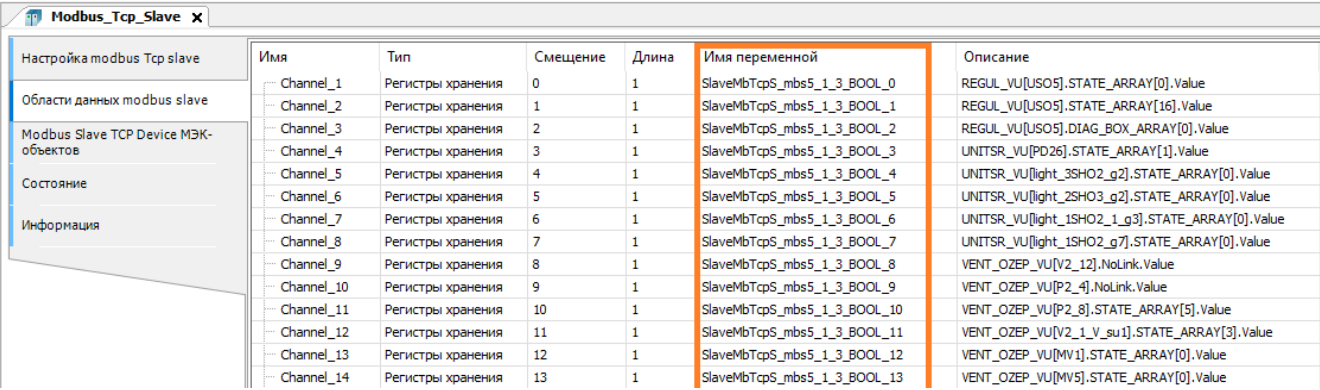
MBS_GVL
1 {attribute 'subsequent' := ''}
2 VAR GLOBAL
3   SlaveMbTcpS_mbs5_1_3_BOOL_0: WORD;
4   SlaveMbTcpS_mbs5_1_3_BOOL_1: WORD;
5   SlaveMbTcpS_mbs5_1_3_BOOL_2: WORD;
6   SlaveMbTcpS_mbs5_1_3_BOOL_3: WORD;
7   SlaveMbTcpS_mbs5_1_3_BOOL_4: WORD;
8   SlaveMbTcpS_mbs5_1_3_BOOL_5: WORD;
9   SlaveMbTcpS_mbs5_1_3_BOOL_6: WORD;
10  SlaveMbTcpS_mbs5_1_3_BOOL_7: WORD;
11  SlaveMbTcpS_mbs5_1_3_BOOL_8: WORD;
12  SlaveMbTcpS_mbs5_1_3_BOOL_9: WORD;
13  SlaveMbTcpS_mbs5_1_3_BOOL_10: WORD;
14  SlaveMbTcpS_mbs5_1_3_BOOL_11: WORD;
15  SlaveMbTcpS_mbs5_1_3_BOOL_12: WORD;
16  SlaveMbTcpS_mbs5_1_3_BOOL_13: WORD;
17  SlaveMbTcpS_mbs5_2_3_DWORD_0: DWORD;
18  SlaveMbTcpS_mbs5_2_3_DWORD_1: DWORD;

```

Рисунок

Названия переменных берутся из модуля MbsTcpSlave файла крейта ЭЛСИ-TMK.xml, при парсинге каналов.

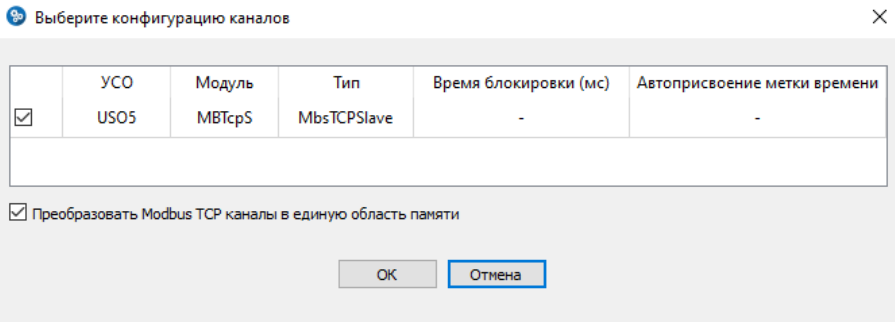
Их же можно найти и в модуле Modbus_Tcp_Slave, каждая переменная привязана к своему каналу:



Имя	Тип	Смещение	Длина	Имя переменной	Описание
Channel_1	Регистры хранения	0	1	SlaveMbTcpS_mbs5_1_3_BOOL_0	REGUL_VU[US05].STATE_ARRAY[0].Value
Channel_2	Регистры хранения	1	1	SlaveMbTcpS_mbs5_1_3_BOOL_1	REGUL_VU[US05].STATE_ARRAY[16].Value
Channel_3	Регистры хранения	2	1	SlaveMbTcpS_mbs5_1_3_BOOL_2	REGUL_VU[US05].DIAG_BOX_ARRAY[0].Value
Channel_4	Регистры хранения	3	1	SlaveMbTcpS_mbs5_1_3_BOOL_3	UNITSR_VU[PD26].STATE_ARRAY[1].Value
Channel_5	Регистры хранения	4	1	SlaveMbTcpS_mbs5_1_3_BOOL_4	UNITSR_VU[light_3SHO2_g2].STATE_ARRAY[0].Value
Channel_6	Регистры хранения	5	1	SlaveMbTcpS_mbs5_1_3_BOOL_5	UNITSR_VU[light_2SHO3_g2].STATE_ARRAY[0].Value
Channel_7	Регистры хранения	6	1	SlaveMbTcpS_mbs5_1_3_BOOL_6	UNITSR_VU[light_1SHO2_1_g3].STATE_ARRAY[0].Value
Channel_8	Регистры хранения	7	1	SlaveMbTcpS_mbs5_1_3_BOOL_7	UNITSR_VU[light_1SHO2_g7].STATE_ARRAY[0].Value
Channel_9	Регистры хранения	8	1	SlaveMbTcpS_mbs5_1_3_BOOL_8	VENT_OZEP_VU[V2_12].NoLink.Value
Channel_10	Регистры хранения	9	1	SlaveMbTcpS_mbs5_1_3_BOOL_9	VENT_OZEP_VU[P2_4].NoLink.Value
Channel_11	Регистры хранения	10	1	SlaveMbTcpS_mbs5_1_3_BOOL_10	VENT_OZEP_VU[P2_8].STATE_ARRAY[5].Value
Channel_12	Регистры хранения	11	1	SlaveMbTcpS_mbs5_1_3_BOOL_11	VENT_OZEP_VU[V2_1_vsu1].STATE_ARRAY[3].Value
Channel_13	Регистры хранения	12	1	SlaveMbTcpS_mbs5_1_3_BOOL_12	VENT_OZEP_VU[MV1].STATE_ARRAY[0].Value
Channel_14	Регистры хранения	13	1	SlaveMbTcpS_mbs5_1_3_BOOL_13	VENT_OZEP_VU[MV5].STATE_ARRAY[0].Value

Рисунок

Если на этапе выбора конфигурации каналов была отмечена опция «Преобразовать Modbus TCP каналы в единую область памяти»



	УСО	Модуль	Тип	Время блокировки (мс)	Автоприсвоение метки времени
<input checked="" type="checkbox"/>	US05	MBTcpS	MbsTCPSlave	-	-

☒ Преобразовать Modbus TCP каналы в единую область памяти

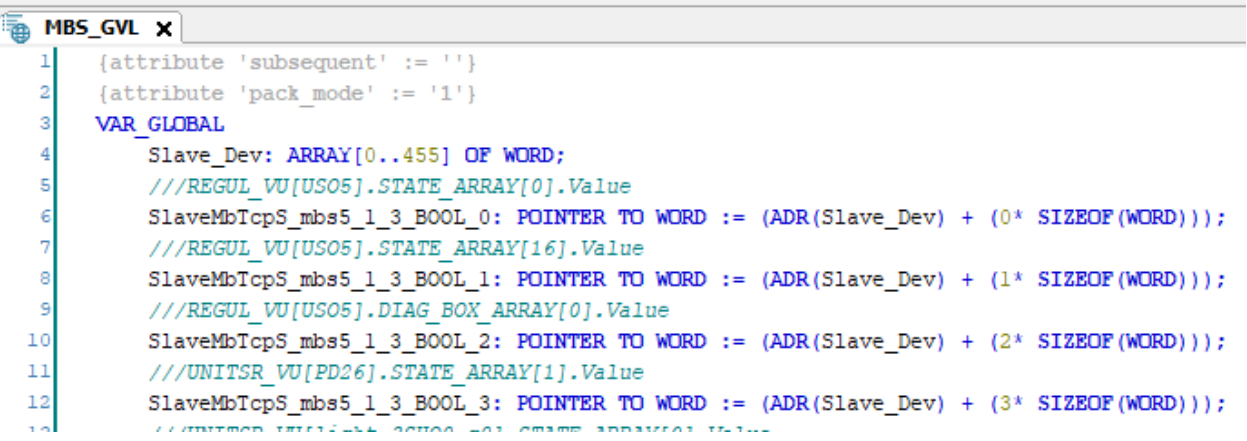
OK Отмена

Рисунок

То созданный файл MBS_GVL помимо атрибута **subsequent**, так же содержит атрибут **pack_mode** со значением 1.

Так же, MBS_GVL содержит в своем заголовке объявление массива Slave_dev такой размерности, которая способна вместить все переменные SlaveMbTcpS_mbs_xxx, привязанные к каналам.

А переменные SlaveMbTcpS_mbs_xxx становятся указателями на массив Slave_Dev.



```

1 {attribute 'subsequent' := ''}
2 {attribute 'pack_mode' := '1'}
3 VAR_GLOBAL
4   Slave_Dev: ARRAY[0..455] OF WORD;
5   ///REGUL_VU[US05].STATE_ARRAY[0].Value
6   SlaveMbTcpS_mbs5_1_3_BOOL_0: POINTER TO WORD := (ADR(Slave_Dev) + (0* SIZEOF(WORD)));
7   ///REGUL_VU[US05].STATE_ARRAY[16].Value
8   SlaveMbTcpS_mbs5_1_3_BOOL_1: POINTER TO WORD := (ADR(Slave_Dev) + (1* SIZEOF(WORD)));
9   ///REGUL_VU[US05].DIAG_BOX_ARRAY[0].Value
10  SlaveMbTcpS_mbs5_1_3_BOOL_2: POINTER TO WORD := (ADR(Slave_Dev) + (2* SIZEOF(WORD)));
11  ///UNITSR_VU[PD26].STATE_ARRAY[1].Value
12  SlaveMbTcpS_mbs5_1_3_BOOL_3: POINTER TO WORD := (ADR(Slave_Dev) + (3* SIZEOF(WORD)));
13  ///UNITSR_VU[light_3SHO2_g2].STATE_ARRAY[0].Value

```

Рисунок

При импорте каналов в модуль Modbus_Tcp_Slave в списке будет всего один канал, связанный с переменной Slave_Dev:

Modbus_Tcp_Slave X						
Настройка modbus Tcp slave	Имя	Тип	Смещение	Длина	Имя переменной	Описание
Области данных modbus slave Data area 1	Регистры хранения	0	456	MBS_GVL.Slave_Dev	United data area

Рисунок

Так же текст внутри всех POU изменяется в связи с добавлением оператора разыменования ^ для указателей из MBS_GVL

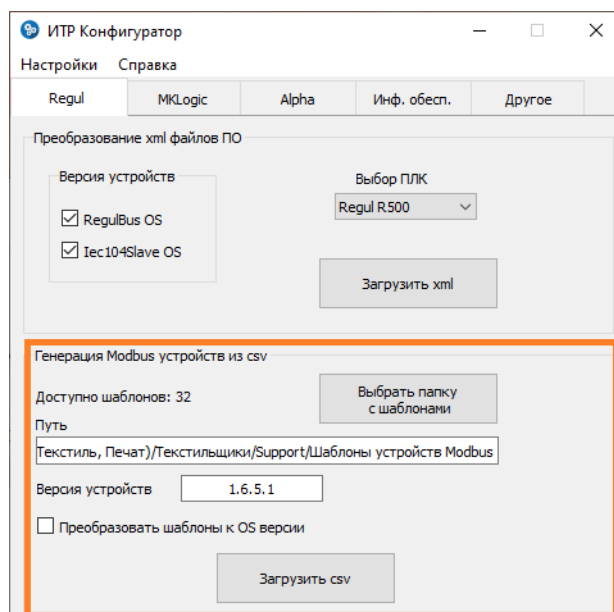
```

US05_IEC_Program_Modbus X
1  PROGRAM US05_IEC_Program_Modbus
2
3  (* Create date: 12.05.2023 8:55:10 *)
4  (* Owner: Собоцкий Александр *)
5  SlaveMbTcpS_mbs5_1_3_BOOL_0^0 := (REGUL_VU[US05].STATE_ARRAY[0].Value);
6
7
8  SlaveMbTcpS_mbs5_1_3_BOOL_0^1 := (REGUL_VU[US05].STATE_ARRAY[1].Value);
9
10
11 SlaveMbTcpS_mbs5_1_3_BOOL_0^2 := (REGUL_VU[US05].STATE_ARRAY[2].Value);
12
13

```

Рисунок

Генерация Modbus устройств из csv

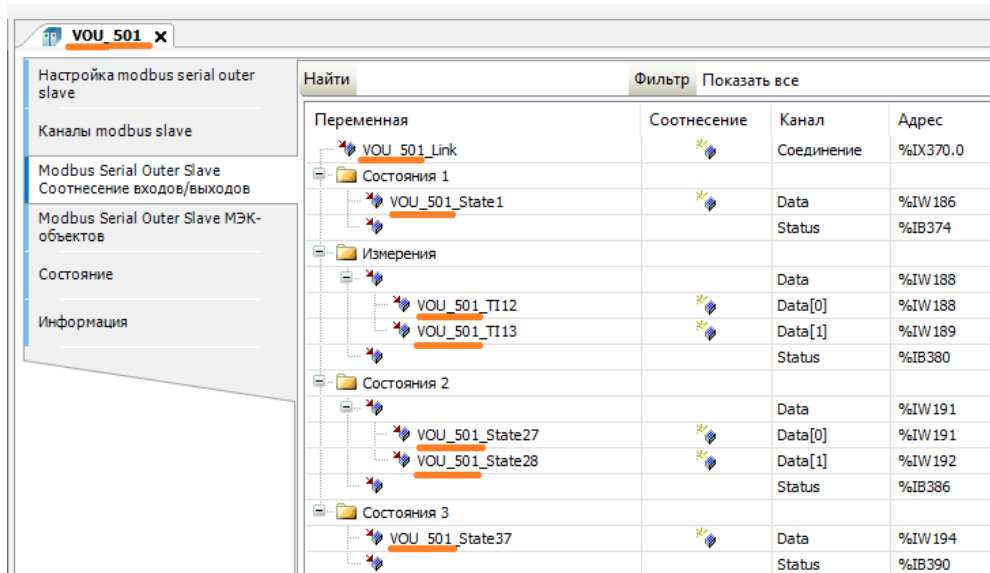


Рисунок

Сначала необходимо выбрать папку с шаблонами. Это директория, содержащая список шаблонов Modbus RTU/TCP устройств.

Новое устройство создается в среде Epsilon LD/Astra IDE, затем экспортируется в виде xml объекта.

Отличительной чертой шаблона является то, что все привязанные переменные начинаются на имя шаблона.



Рисунок

Затем при нажатии на кнопку Загрузить csv можно загрузить один или несколько csv файлов Infinity Server.csv, созданных в программе ACS Configurator.

Конвертер пройдет по всем объектам дерева Infinity Server в поисках следующих свойств:

- Тип Modbus (8814)
- Адрес КП (8888)
- Modbus модуль (8810)
- Modbus канал (8811)
- Маппинг (8812)
- IP адрес (8813)
- Шаблон устройства (8821)
- Modbus_DelayBetweenPolls (89101)

Для каждого объекта проверяется валидность свойств. Если не было заполнено (найдено) хоть одно из необходимых свойств, то генерация modbus устройств по такому объекту не производится.

Для устройств с Тип Modbus равным RTU (0) необходимые поля:

- Тип Modbus
- Маппинг
- Шаблон устройства
- Адрес КП
- Modbus модуль
- Modbus канал

Для устройств с тип Modbus равным TCP (1) необходимые поля:

- Тип Modbus
- Маппинг
- Шаблон устройства
- Адрес КП
- IP адрес

Ниже приведен пример RTU объекта Infinity Server:

№	Атрибут свойства	Значение	
1	IP-адрес		
2	Modbus Master Контроллер	REGUL	
3	Modbus канал	3	
4	Modbus канал 2	<[1..4]>	
5	Modbus модуль	8	
6	Modbus модуль 2		
7	Адрес КП	250	
8	Адрес КП 2	<[1..254]>	
9	Количество ТИ	<[0..255]>	
10	Количество ТР	0	
11	Количество ТС	<[0..255]>	
12	Количество ТУ	0	
13	Маппинг	mbs_n_SU_12215p_Ch1	
14	Маппинг 2		
15	резерв	нет	
16	тип	vt_bstr	
17	Тип Modbus	RTU	
18	Шаблон устройства	VOU_501	
19	Шаблон устройства 2		

Рисунок

Для каждого RTU или TCP объекта Infinity Server будет произведена проверка наличия указанного .xml файла шаблона устройства в выбранной папке с шаблонами (в нашем примере наличие файла VOU_501.xml).

Если шаблона нет в папке, объект пропускается. В конце программа выведет список отсутствующих шаблонов, если такие имеются.

После того, как программа пройдет по всем объектам Infinity Server и составит список объектов, по которым нужно сгенерить Modbus устройства, будет произведено удаление найденных дубликатов среди списка объектов.

Например, несколько объектов Infinity Server могут забирать информацию с одного и того же устройства modbus. Например, в случае некоторых клапанов, в ACS Configurator'e каждый клапан представляет собой отдельный объект Infinity Server, но modbus устройство на все клапана – одно.

Дубликатом будут считаться объекты в следующих случаях:

а). Если тег объекта начинается на USOX., то дубликатами считаются объекты с одного УСО и одинаковыми значениями поля Маппинг

б). Или если тег объекта не начинается на USOX, то дубликатом считаются объекты из одного и того же .csv файла с одинаковым значением поля Маппинг.

Затем программа проходится по каждому объекту, берет соответствующий шаблон, модифицирует его и сохраняет в папку modules для RTU шаблонов, и в папку modules_tcp для TCP шаблонов.

В папке modules_tcp содержатся все TCP шаблоны по одиночке для каждого найденного устройства, а также файл DEV_ALL.xml, который содержит внутри себя все шаблоны.

В папке modules все RTU шаблоны располагаются внутри вложенных папок согласно номеру модуля и номеру канала.

Автоматические python скрипты импортируют файл DEV_ALL.xml.

Модификация Modbus шаблона включает этапы:

- Замена имени шаблона – в нашем примере имя шаблона VOU_501 поменяется на mbs_n_SU_12215p_Ch1 как в имени modbus устройства, так и во всех его внутренних переменных. Например, вместо VOU_501_Link привязка будет mbs_n_SU_12215p_Ch1_Link.
- Замена адреса КП – число из свойства «Адрес КП» попадает в графу «Адрес ведомого устройства».
- Замена IP адреса – текст из свойства «IP адрес» попадает в графу «IP адрес»
- Замена версии устройства – текст из поля «Версия устройств» конвертера заменяется внутри узла Version шаблона
- Замена типа устройства – в зависимости от отмеченной опции «Преобразовать шаблоны к OS версии», результирующее устройство будет либо OS версии, либо нет. Сам шаблон при этом может быть как OS-ный, так и обычный.

Если Infinity Server объект содержал так же свойство «Modbus_DelayBetweenPolls» со значением 1, то в файле DEV_ALL.xml после этого объекта вставится пустышка oven_dummy.xml с адресом КП = 247 и одним каналом

по нулевому адресу. Это своего рода костыль для ОВНов, которые то ли неправильно отвечают на запросы, то ли что-то с ними еще не так и они пропадают со связи. С пустышками становится вроде лучше.

Вообще, каждый Infinity Server объект может содержать до 5 разных шаблонов устройств. Например, если необходимо несколько каналов для одних и тех же данных (шахты, ВОУ). Для каждого нового шаблона имеется свой набор свойств:

Шаблон 1:

- Адрес КП (свойство 8888)
- Modbus модуль (свойство 8810)
- Modbus канал (свойство 8811)
- Маппинг (свойство 8812)
- IP адрес (свойство 8813)
- Шаблон устройства (свойство 8821)

Шаблон 2:

- Адрес КП 2 (свойство 8816)
- Modbus модуль 2 (свойство 8817)
- Modbus канал 2 (свойство 8818)
- Маппинг 2 (свойство 8819)
- IP адрес 2 (свойство 8820)
- Шаблон устройства 2 (свойство 8822)

Шаблон 3

- Адрес КП 3 (свойство 8827)
- Modbus модуль 3 (свойство 8828)
- Modbus канал 3 (свойство 8829)
- Маппинг 3 (свойство 8830)
- IP адрес 3 (свойство 8831)
- Шаблон устройства 3 (свойство 8832)

Шаблон 4:

- Адрес КП 4 (свойство 8833)
- Modbus модуль 4 (свойство 8834)
- Modbus канал 4 (свойство 8835)
- Маппинг 4 (свойство 8836)
- IP адрес 4 (свойство 8837)
- Шаблон устройства 4 (свойство 8838)

Шаблон 5:

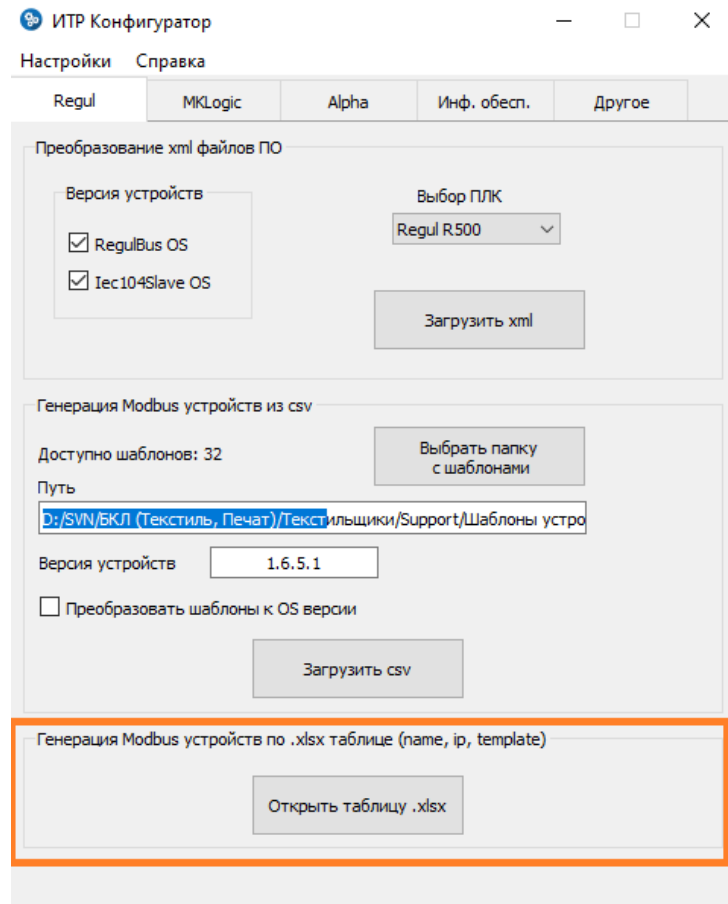
- Адрес КП 5 (свойство 8911)
- Modbus модуль 5 (свойство 8912)
- Modbus канал 5 (свойство 8913)
- Маппинг 5 (свойство 8914)
- IP адрес 5 (свойство 8915)
- Шаблон устройства 5 (свойство 8916)

Следующие свойства могут присутствовать в ACS Конфигураторе, но не берутся во внимание конвертером:

- Modbus Master Контроллер

- Количество ТИ
- Количество ТР
- Количество ТС
- Количество ТР

Генерация Modbus устройств по .xlsx таблице



Рисунок

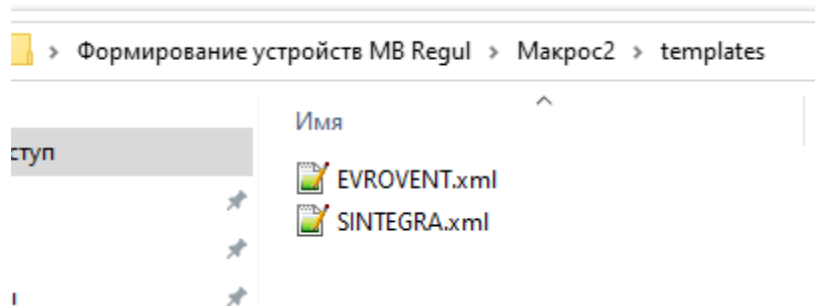
Этот вариант используется только на депо Сокол.
Необходимо выбрать таблицу .xlsx такого формата:

	A	B	C
1	SHUPV1	192.168.1.1	EVROVENT
2	SHUV2	192.168.1.2	SINTEGRA
3			
4			
5			

Рисунок

Где в первой колонке пишем имя устройства, во второй колонке его IP адрес, а в третьей колонке его имя шаблона.

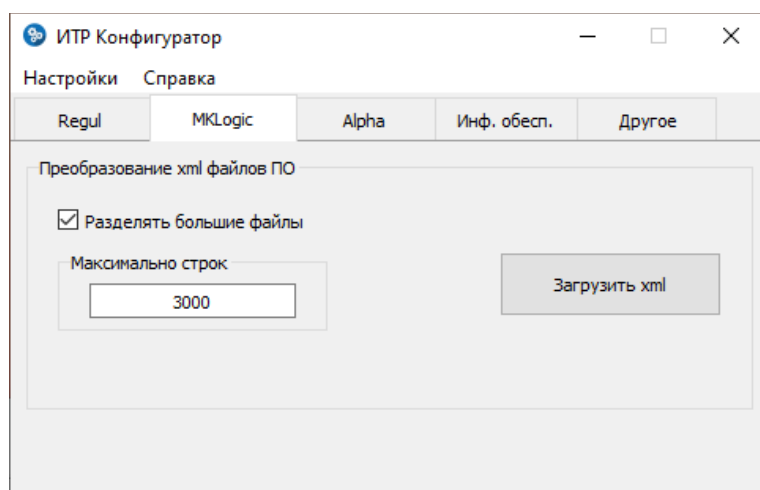
На одном уровне с таблицей должна быть папка templates с шаблонами:



Рисунок

Конвертер просто пойдет по списку, откроет нужный шаблон, поменяет там IP адрес и имя и сохранит в предложенном месте все табличные устройства.

Вкладка MKLogic



Рисунок

Нажатие кнопки «Загрузить XML» открывает диалоговое окно для выбора файлов ЭЛСИ-TMK_Application.xml, которые предварительно были созданы программой ACS Configurator.

Работа с крейтом не производится.

Цель – модификация приложения контроллера.

Для каждого выбранного Application.xml файла создается одноименная папка, куда попадают выходные модифицированные конвертером файлы.

Этапы модификации приложения:

1. Создание .csv файла привязки переменных
2. Модификация POU

Создание .csv файла привязки переменных

Программа находит все списки глобальных переменных (global_vars, const_vars и др.) и переносит все переменные согласно их типу в .csv файл такого формата, который может быть импортирован в среду isagraf.

Каждой переменной создается строка типа:

"%1,%2,%3,,%4,Var,ReadWrite,,,,False,,"

Где в плейсхолдеры %1-%4 – подставляются значения:

%1 – имя переменной

%2 – тип переменной

%3 – для массивов размерность, для остальных типов – ничего

%4 – начальное значение (если имеется)

Модификация POU

Программа проходит по всем объектам POU, которые имеют тип программа (program), т.е. имеют какой-то ST текст внутри.

Забирает внутренний ST текст каждого POU, модифицирует его и сохраняет отдельными .txt файлами каждую секцию.

Секция USO1_sFB становится USO1_FB.txt.

Секция USO1_sFBModbus становится USO1_FBModbus.txt.

Секция USO1_IEC_870_5 становится USO1_lecProgram.txt.

Секция USO1_IEC_870_5_ED становится USO1_lecProgram_EDC.txt.

В зависимости от настройки «Разделять большие файлы» в совокупности с максимальным числом строк, каждая из секций может быть сохранена в несколько файлов, каждый из которых будет содержать количество строк, не превышающих указанное число.

Этапы модификации ST текста каждого POU:

1. Модификация индексов IEC
2. Замена циклов FOR на WHILE
3. Модификация ФБ

Так же, любая пустая строка, не содержащая текста, пропускается, в результате чего выходные .txt файлы не содержат пробелов между строками.

Модификация индексов IEC

В секциях IEC_870_5 преобразуем следующий блок кода:

```
indexIEC := 0;
IF indexIEC < 512 THEN
    IEC104_SP_Out30_1[indexIEC].value := (CrOldIsDsp);
ELSIF indexIEC < 1024 THEN
    IEC104_SP_Out30_2[indexIEC - 512].value := (CrOldIsDsp);
ELSIF indexIEC < 1536 THEN
    IEC104_SP_Out30_3[indexIEC - 1024].value := (CrOldIsDsp);
ELSIF indexIEC < 2048 THEN
    IEC104_SP_Out30_4[indexIEC - 1536].value := (CrOldIsDsp);
ELSIF indexIEC < 2560 THEN
    IEC104_SP_Out30_5[indexIEC - 2048].value := (CrOldIsDsp);
ELSE
    IEC104_SP_Out30_6[indexIEC - 2560].value := (CrOldIsDsp);
END_IF;
```

Рисунок

В одну строку в зависимости от значения переменной indexIEC.

Например, для indexIEC := 511 строка будет такой:

IEC104_SP_Out30_1[511].value := (CrOldIsDsp);

А для индекса 512 уже будет так:

IEC104_SP_Out30_2[0].value := (CrOldIsDsp);

Помимо строчек с .value в условиях так же может быть указан флаг качества .IV:

```

indexIEC := 178;
IF indexIEC <= 512 THEN
    IEC104_SP_Out30_1[indexIEC].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_1[indexIEC].IV :=VouVu[su12248p].State[14].IV;
ELSIF indexIEC <= 1024 THEN
    IEC104_SP_Out30_2[indexIEC - 512].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_2[indexIEC - 512].IV :=VouVu[su12248p].State[14].IV;
ELSIF indexIEC <= 1536 THEN
    IEC104_SP_Out30_3[indexIEC - 1024].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_3[indexIEC - 1024].IV :=VouVu[su12248p].State[14].IV;
ELSIF indexIEC <= 2048 THEN
    IEC104_SP_Out30_4[indexIEC - 1536].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_4[indexIEC - 1536].IV :=VouVu[su12248p].State[14].IV;
ELSIF indexIEC <= 2560 THEN
    IEC104_SP_Out30_5[indexIEC - 2048].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_5[indexIEC - 2048].IV :=VouVu[su12248p].State[14].IV;
ELSE
    IEC104_SP_Out30_6[indexIEC - 2560].value :=(VouVu[su12248p].State[14].Value);
    IEC104_SP_Out30_6[indexIEC - 2560].IV :=VouVu[su12248p].State[14].IV;
END_IF;

```

Рисунок

Тогда на выходе будет две строчки:

```

IEC104_SP_Out30_1[178].value :=(VouVu[su12248p].State[14].Value);
IEC104_SP_Out30_1[178].IV :=VouVu[su12248p].State[14].IV;

```

Замена циклов FOR на WHILE

Все циклы FOR заменяется на аналогичные WHILE. Переменная Index меняется на indexFB везде, кроме секции sFBModbus (там она меняется на indexFBModbus).

Было:

```

(* execution typeAvrFB *)
FOR Index := avr1C_21 TO c6AI2 DO
    typeAvrFB (Vu := AvrVu[Index],
              Prg := AvrPrg[Index],
              Dio := AvrDio[Index]);
END_FOR;

```

Рисунок

Стало:

```

(* execution typeAvrFB *)
indexFB := avr1C_21;
WHILE indexFB <= c6AI2 DO
    tmpBool := typeAvr (indexFB);
    indexFB := indexFB + 1;
END_WHILE;

```

Рисунок

Обратите внимание, что вызов ФБ в цикле тоже изменился, но это уже следующий этап.

Когда программа находит ФБ, она его видит в формате

```
nameFB (arg1 := arg2[arg3], arg1 := arg2[arg3]...)
```

Все преобразование сводится к тому, что нам нужно вызвать этот же ФБ и сохранить результат в переменную tmpBool.

Мы всегда из имени FB удаляем в конце символы FB

Варианты модификаций:

1. Если ФБ вызывается без аргументов:

```
testFB();
```

То он будет заменен на такую строчку:

```
tmpBool := test ();
```

2. Если имя ФБ НЕ содержит typeModbus, например:

```
(* ст. Аминьевская БЭ №1 Отсутствие связи *)
typeLink2FB (Vu := LinkVu[Be1_NoLink],
             Prg := LinkPrg[Be1_NoLink],
             Dio := LinkDio[Be1_NoLink]);
```

Рисунок

То мы вызываем ФБ и единственным его аргументом становится индекс:

```
(* ст. Аминьевская БЭ №1 Отсутствие связи *)
tmpBool := typeLink2 (Be1_NoLink);
```

Рисунок

То есть:

Было

```
typeLink2FB (arg1 := arg2[Index], arg1 := arg2[Index]...)
```

Стало

```
tmpBool := typeLink2 (Index);
```

3. Если имя ФБ содержит typeModbus, например

```
(* ст. Аминьевская ШКМ №1 Овен MB110-224.8A *)
typeModbusOven8AIFB (cfgDevModbus := OvencfgDevModbus[schkm1_Modbus]);
```

Рисунок

То мы преобразовываем это к вызову ФБ без имен аргументов, только сами аргументы.

```
(* ст. Аминьевская ШКМ №1 Овен MB110-224.8A *)
tmpBool := typeModbusOven8AI (OvencfgDevModbus[schkm1_Modbus]);
```

Рисунок

То есть:

Было

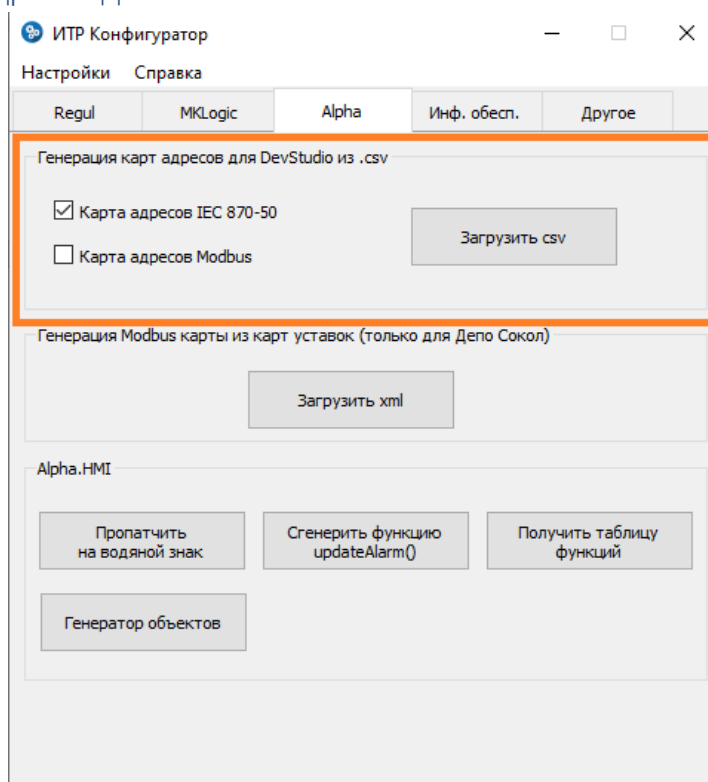
```
typeModbusTestFB (arg1 := arg2[arg3], arg1 := arg2[arg3]...)
```

Стало

```
tmpBool := typeModbusTest (arg2[arg3], arg2[arg3] ...);
```

Вкладка Alpha

Генерация карт адресов для DevStudio из csv



Рисунок

Рассматриваемое действие позволяет из .csv файлов дерева сигналов Infinity Server'a, экспортируемого программой ACS Configurator, создать карты привязки сигналов к адресам для программы DevStudio.

В зависимости от выбранной настройки IEC 870-5 или Modbus программа будет создавать привязку адресов либо на МЭК протокол, либо на Modbus.

Для правильной работы программы, в ACS Configurator должны быть проставлены все связи, что бы теги Infinity Server были адресные.

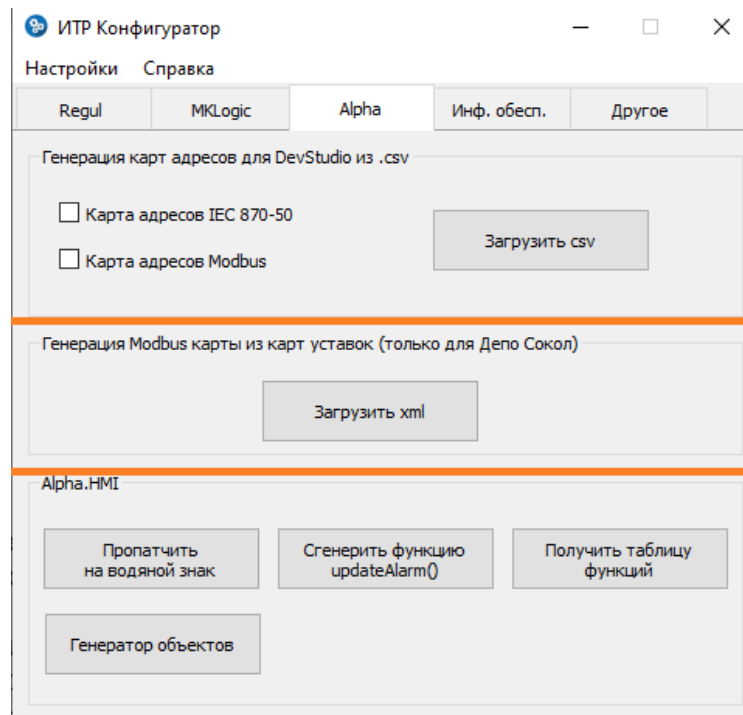
Можно загружать несколько .csv файлов, чтобы на выходе получить несколько карт адресов. Причем не обязательно иметь несколько .csv файлов по одному на каждую КП (modbus) или станцию (МЭК). Можно иметь один общий .csv файл с различными номерами КП и номерами станций внутри.

Карты адресов Modbus сохраняются в файл Modbus_TCP_USO?.xml, где ? – номер КП.

Карты адресов МЭК сохраняются в файл IEC104_USO?.xml, где ? – номер станции.

Генерация Modbus карты из карт уставок

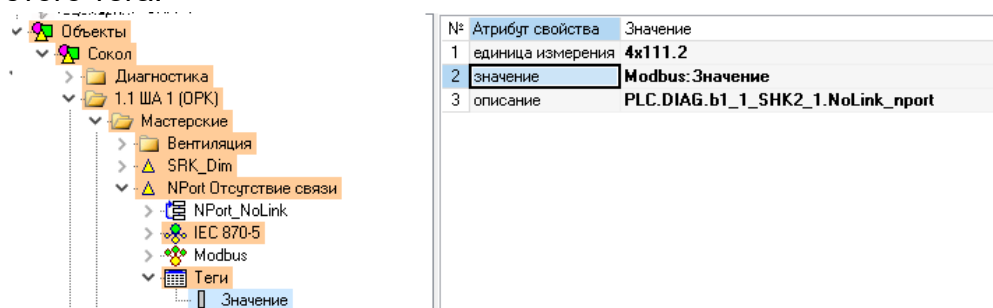
Данный функционал используется только для депо Сокол, для остальных проектов и Депо – используется генерация карт адресов из .csv.



Рисунок

Кнопка Загрузить xml позволяет загрузить одну или несколько карт уставок из программы ACS Configurator проекта Депо Сокол. И на выходе получить файлы для DevStudio с заполненными адресами Modbus.

В программе ACS конфигуратор каждый тег в карте уставок объекта содержит поле описание, где храниться тег и поле единица измерения где хранится адрес у этого тега:



Рисунок

Результирующий .xml файл карт Modbus адресов будет называться так же, как и карта уставок, только с заменой слова «Теги» на «MAP».

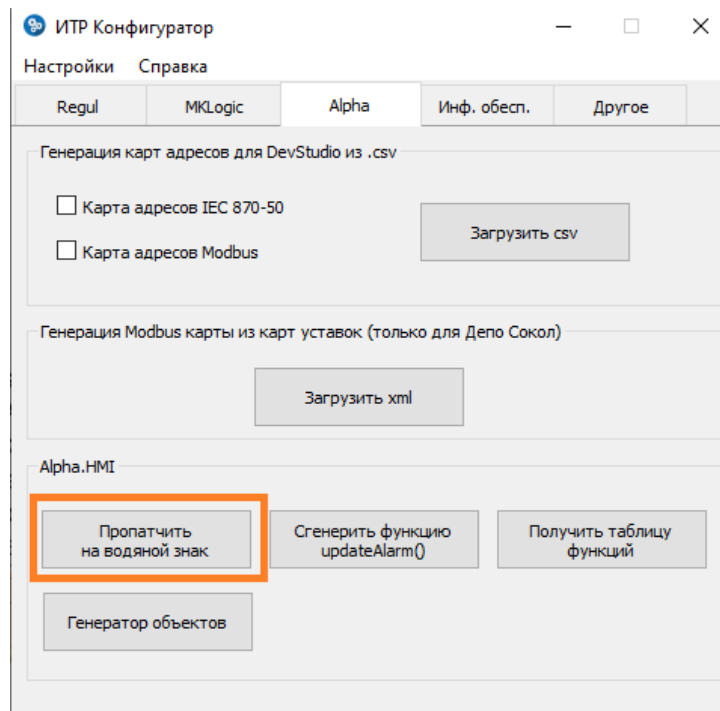
Alpha.HMI

Пропатчить на водяной знак

Программа предлагает выбрать файл библиотеки **alpha.hmi.ui.dll** для Windows или **libalpha.hmi.ui.so** для Linux.

Данный файл лежит папке вместе с установленными бинарниками Alpha.HMI.

Пропатчив его, исчезнет водяной знак на HMI формах, появляющийся после 15-ти совокупных минут работы формы в режиме исполнения.



Рисунок

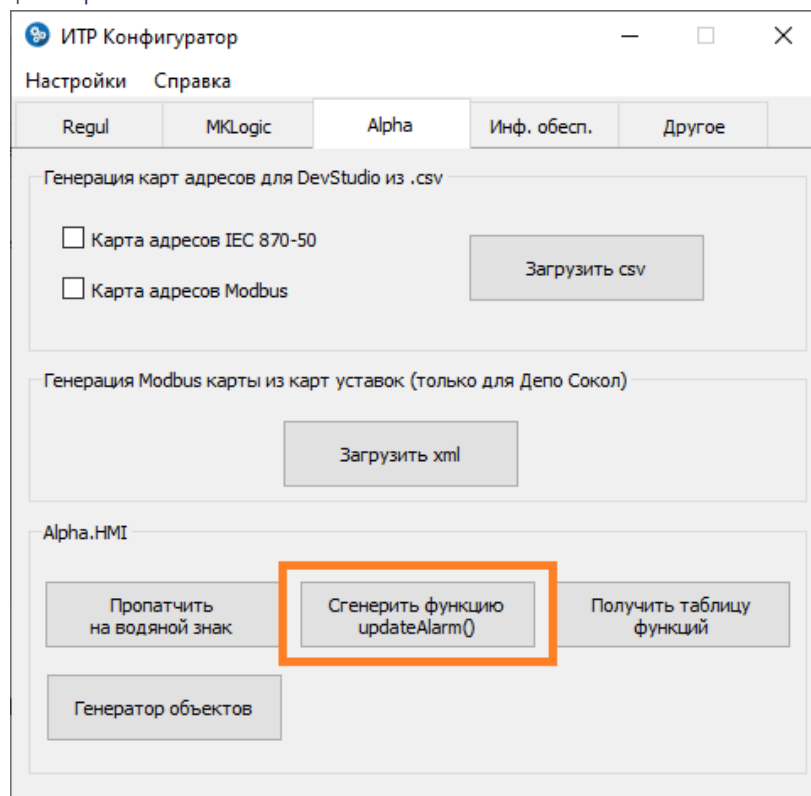
Конвертер должен пропатчить загруженную библиотеку на предмет замены четырех строчек на пробелы:

- Лицензия не обнаружена или ограничена.
- Использование %1 без лицензии допустимо только для разработки и испытания проектов автоматизации инжиниринговыми компаниями.
- Не допускается использование нелицензированных компонентов конечными пользователями на объектах.
- Все права защищены © %1

Первая строка - 71 пробел, вторая - 232, третья - 183, четвертая - 40. На все русские символы - по два пробела, на другие - один.

Т.е. мы просто скрываем водяной знак, сама его отрисовка воспроизводится и в больших проектах может все так же чуть снижать производительность.

Сгенерить функцию updateAlarm



Рисунок

Данная опция предлагает выбрать файл SETTINGS.xml, который лежит в папке support внутри любого Alpha.HMI проекта ЭМС, Депо.

Данный файл содержит внутри себя список экранов проекта и список типов элементов. У каждого экрана есть свой код. У каждого типа элемента есть свой аварийный тег.

Конвертер проходит по списку экранов, открывает в режиме чтения каждый .omobj экран из папки objects HMI проекта, смотрит имеющиеся внутри объекты и их типы. Если найденный объект по типу совпадает с описанным типом внутри файла SETTINGS.xml, то с объекта забирается алиас, а с типа – аварийный тег, таким образом формируя полный аварийный тег до конкретного объекта.

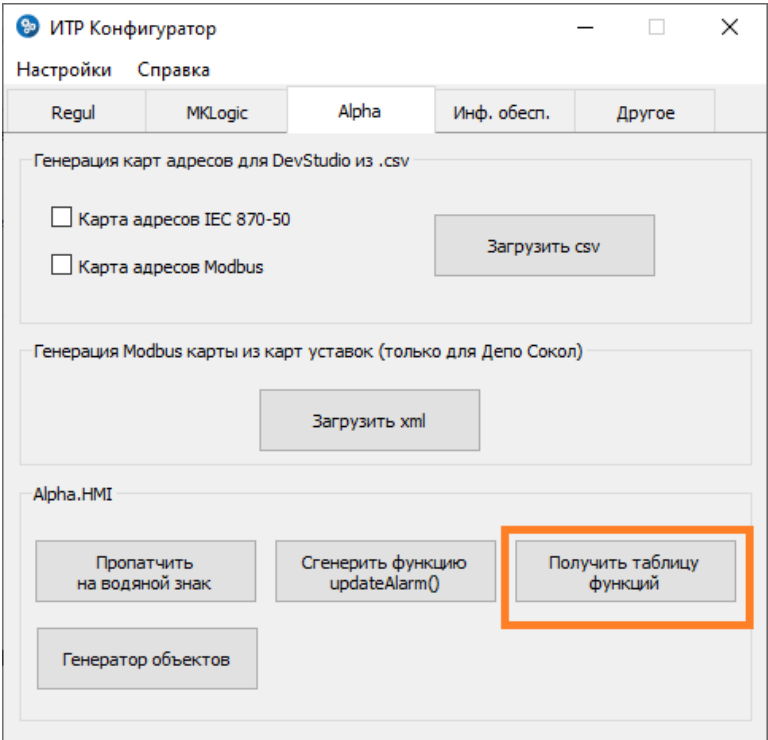
В итоге получаем актуальные списки аварийный тегов объектов по каждому экрану. И javascript функцию, которая возвращает нужный список по коду экрана.

Дальше уже внутри HMI проекта эту функцию вызывают верхние кнопки навигации, передавая свой код, чтобы по аварийным тегам загораться красным.

Более подробно про формат файла SETTINGS.xml можно прочитать внутри самого файла SETTINGS.xml.

Программа сохраняет содержимое функции внутри файла func.txt и предлагает сразу же обновить содержимое функции в самом HMI проекте. Если отказаться от автоматической замены, то все содержимое файла func.txt нужно будет потом вручную вставить по пути unit.Global.updateAlarm().

Получить таблицу функций



Рисунок

Данная кнопка позволяет загрузить все тот же файл SETTINGS.xml, но важен будет лишь список экранов.

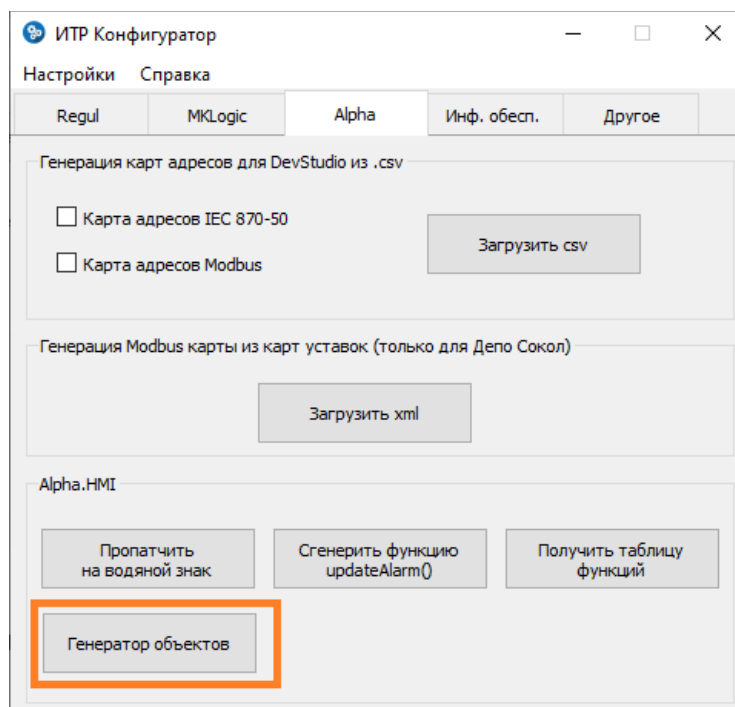
Программа откроет каждый экран проекта, указанный в файле SETTINGS.xml и создаст файл func_table.xlsx - таблицу объектов и их функций (поле in_FUNCTION) для построения попапов.

Таким образом можно быстро будет понять, какой объект использует какую функцию для построения своего попапа.

	A	B	C	D	E
1	Имя	Тип	Ter	Функция	Экран
2	n_SU_12215p	VOU1	.USO1.n_SU_12215p	SU1	s_main
3	n_SU_12215	VOU2	.USO1.n_SU_12215	SU2	s_main
4	n_SU_12214p	VOU1	.USO2.n_SU_12214p	SU1	s_main
5	n_SU_12214	VOU2	.USO2.n_SU_12214	SU2	s_main
6	n_OVU_12182	OVU	.USO2.n_OVU_12182	OVU3	s_main
7	n_MVU_12183	VOU2	.USO4.n_MVU_12183	MVU2	s_main
8	n_MVU_12184	VOU2	.USO1.n_MVU_12184	MVU2	s_main
9	n_SU_12216p	VOU1	.USO2.n_SU_12216p	SU1	s_main
10	n_SU_12216	VOU2_SK	.USO2.n_SU_12216	SU2_SK	s_main
11	n_SU_12217p	VOU1	.USO3.n_SU_12217p	SU1	s_main
12	n_SU_12217	VOU2_SK	.USO3.n_SU_12217	SU2_SK	s_main

Рисунок

Генератор объектов



Рисунок

Данная опция позволяет создавать типовые объекты на HMI форме и расключать их на нужные теги.

Для этого нам нужно выбрать сначала .xml файл с типами, затем .csv файл с объектами.

XML файл с типами имеет формат:

```
<types>
  <object ...
  <object ...
  ...
</types>
```

Для его создания для начала создайте в Alpha.HMI любую форму, затем перетащите нужные мнемознаки на форму и расключите их свойство in_Alias на какой-то алиас, например на:

```
unit.Global.RootTag + ".USO1.Test"
```

После этого найдите созданную сохраненную форму внутри папки objects HMI проекта (она будет иметь расширение .omobj), откройте ее текстовым редактором и найдите узлы <object ... соответствующие вашим добавленным объектам.

Скопируйте все нужные узлы <object ... внутрь узлов <types> документа types.xml.

На этом создание XML файла с типами завершено.

Пример узла object мнемознака SHU_V:

```
<object access-modifier="private" name="SHU_V" display-name="SHU_V" uuid="68157a60-fd21-4f20-9957-6393dc4dcc8f" base-type="SHU_V" ...  
...  
</object>
```

Следом нам нужно создать .csv файл формата:

<Тип объекта, Имя, Тег, Расположение (помещение)>

Например:

```
SHU_V,ША-П4; B4.1, .US02.Vent.P4_V4_1,Венткамера (7019)  
SHU_V,ША-П2; B2.1; B2.2; B4.3, .US02.Vent.P2_V2_1_V2_2_V4_3,Венткамера (7019)  
SHU_V,ША-B1.1, .US02.Vent.V1_1,Венткамера (7019)  
...
```

Обратите внимание, что csv формат (csv – comma separated values), отделяет атрибуты строки запятыми. Поэтому у вас не должно быть никаких запятых внутри названий помещений или имени установки.

Конвертер создаст на выходе файл FORM_BIG.omobj, являющийся формой HMI, на которую поместит все объекты из csv файла.

Первый аргумент «Тип объекта» соотносится с названием нужного типа из документа types.xml. Соответственно из types.xml просто копируется нужный узел object и вставляется на форму FORM_BIG.omobj.

Затем внутри объекта произойдет замена свойства in_Alias на третий аргумент csv строки – «Тег». Замена происходит по следующему алгоритму:

Если найден знак + в строке in_Alias объекта, то заменяется весь текст справа после + (потому что обычно алиас выглядит как unit.Global.RootTag + ".USOX.Obj.Tag"). Если знака + нет, то меняется полностью все поле на «Тег».

Программа постарается размещать все объекты на форме внутри прямоугольников-комнат, сортируя все объекты по расположению.

Если среди всех csv строк будет хоть одна строка со скобочками (), то программа будет сортировать объекты строго по номеру помещений. Если же не указывать помещения через скобочки, то сортировка будет просто по текстовому расположению.

Если у объекта будет внутри вложенный текстовый объект "Name", то в его значение подставится «Имя» из второго аргумента csv файла. Обычно для Депо используется.

И последнее.

Если внутри файла с типами types.xml будет хотя бы один тип с названием NoLink, то все объекты из csv файла независимо от их поля «Тип объекта» выгрузятся на форму с типом NoLink.

Вкладка Инф. обесп.

ИТР Конфигуратор

Настройки Справка

Regul MKLogic Alpha Инф. обесп. Другое

Парсинг адресных свойств

☐ Брать из свойства 7050

☒ Брать из св-в 5564-5567

Загрузить csv файл(ы)

Номер КП	Сигналы	В резерве	Пустые	Наложений адресов
----------	---------	-----------	--------	-------------------

Настройки генерации ИО

☒ Не включать "Резерв"

☒ Не включать Пустые

☒ Подсвечивать наложения

Сгенерировать ИО

Создать txt файл описаний для МЭК выюшки

Рисунок

Данная вкладка позволяет по .csv файлам Infinity Server создать файл информационного обеспечения (ИО) для протокола МЭК-104.

Можно получить файл ИО двумя способами, исходя из выбора настройки Парсинг адресных свойств:

Парсинг адресных свойств

☒ Брать из свойства 7050

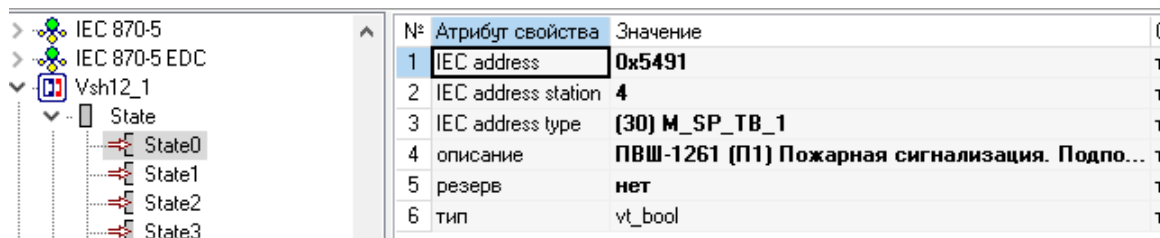
☐ Брать из св-в 5564-5567

Рисунок

Если выбрана опция «Брать из свойства 7050», то конвертер возьмет атрибуты ProtocolType, Address, Station из свойства 7050. Данное свойство (7050) появляется, если в программе ACS Configurator создать связи между отображениями Infinity Server и IEC 870-5.

Если же выбрана опция «Брать из св-в 5564-5567», то необходимо везде, где необходимо, добавить свойства для тегов отображения Infinity Server:

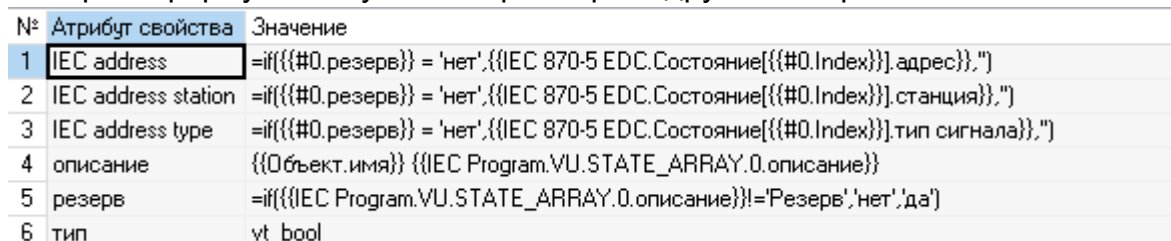
- IEC address (5564)
- IEC address type (5565)
- IEC address station (5567)



№	Атрибут свойства	Значение	
1	IEC address	0x5491	Т
2	IEC address station	4	Т
3	IEC address type	{30} M_SP_TB_1	Т
4	описание	ПВШ-1261 (П1) Пожарная сигнализация. Подпо...	Т
5	резерв	нет	Т
6	тип	vt_bool	Т

Рисунок

И собрать формулами нужные параметры с других отображений:

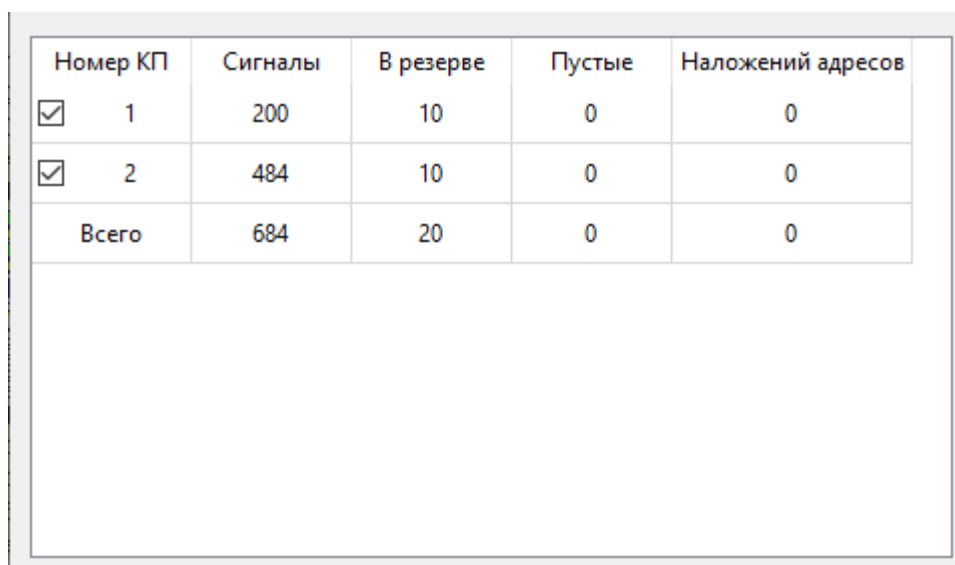


№	Атрибут свойства	Значение
1	IEC address	=if({{#0.резерв}} = 'нет', {{IEC 870-5 EDC.Состояние[{{#0.Index}}].адрес}}, "")
2	IEC address station	=if({{#0.резерв}} = 'нет', {{IEC 870-5 EDC.Состояние[{{#0.Index}}].станция}}, "")
3	IEC address type	=if({{#0.резерв}} = 'нет', {{IEC 870-5 EDC.Состояние[{{#0.Index}}].тип сигнала}}, "")
4	описание	{{Объект.имя}} {{IEC Program.VU.STATE_ARRAY.0.описание}}
5	резерв	=if({{IEC Program.VU.STATE_ARRAY.0.описание}} != 'Резерв', 'нет', 'да')
6	тип	vt_bool

Рисунок

Описание сигнала берется со свойства «описание» не зависимо от выбранной настройки «Парсинг адресных свойств».

После загрузки .csv файлов в программу, отобразится таблица найденных сигналов:



	Номер КП	Сигналы	В резерве	Пустые	Наложений адресов
<input checked="" type="checkbox"/>	1	200	10	0	0
<input checked="" type="checkbox"/>	2	484	10	0	0
	Всего	684	20	0	0

Рисунок

Чекбокс напротив номера КП позволяет убрать или добавить станцию в результирующий файл ИО.

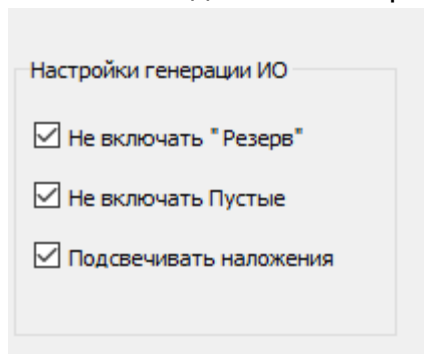
Колонка «Сигналы» показывает общее количество найденных адресных сигналов.

Колонка «В резерве» показывает сколько сигналов имеют описание либо равным слову «Резерв» либо оканчивающиеся на « Резерв».

Колонка «Пустые» показывает сколько сигналов имеют незаполненное поле «описание».

Поле «Наложений адресов» показывает, сколько сигналов внутри данного номера КП (станции) имеют одинаковый адрес и тип протокола.

После этого можно отметить необходимые «Настройки генерации ИО»:



Рисунок

И нажать на кнопку «Сгенерировать ИО». Программа предложит сохранить файл МЭК-104 Инф. обеспечение.xlsx. Который выглядит следующим образом:

	A	B	C	D	E	F
1	Описание	Станция (КП)	Тип (код)	Тип (название)	Адрес (dec)	Адрес (hex)
2	Пожар (платформа)	1	30	M_SP_TB_1	21759	0x54ff
3	Пожар (кроме платформы)	1	30	M_SP_TB_1	21760	0x5500
4	Пожар (перегон 1)	1	30	M_SP_TB_1	21761	0x5501
5	Пожар (перегон 2)	1	30	M_SP_TB_1	21762	0x5502
6	Пожар (II главный путь от ПК114+69 до ПК143+74)	1	30	M_SP_TB_1	21763	0x5503
11	1С-25 АВР 2 Линейное напряжение Ubc	1	36	M_ME_TF_1	27660	0x6c0c
12	1С-25 АВР 2 Линейное напряжение Uca	1	36	M_ME_TF_1	27661	0x6c0d

Рисунок

По каждому КП создается своя вкладка УСО-<номер КП>, куда попадают адресные сигналы.

Если была выбрана настройка «Не включать Резерв», в файле ИО будут отсутствовать строчки, которые по таблице – В резерве. То же самое с настройкой «Не включать Пустые».

Если было выбрано «Подсвечивать наложения», то строчки внутри одного листа (номера КП), имеющие одинаковые адрес и тип протокола, будут подсвечены желтым цветом.

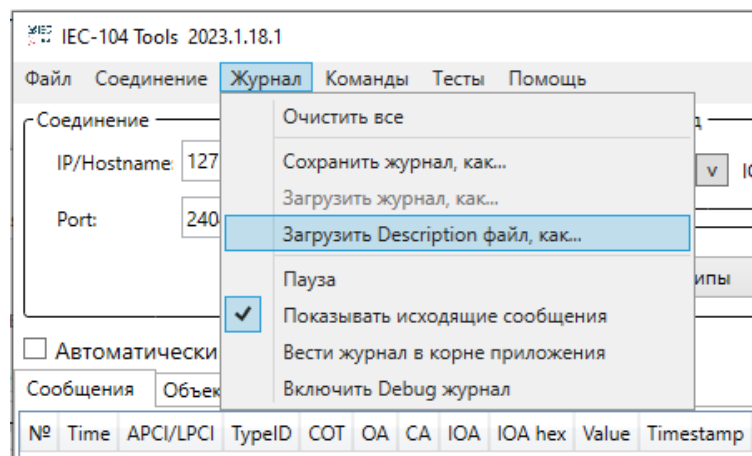
Нажав на кнопку «Создать txt файл описаний для МЭК выюшки», программа создаст файл fileDescription.txt формата

номер станции; адрес (десятич); описание
--

fileDescription.txt	
1	1;21759;Пожар (платформа)
2	1;21760;Пожар (кроме платформы)
3	1;21761;Пожар (перегон 1)
4	1;21762;Пожар (перегон 2)

Рисунок

Данный файл можно затем импортировать в программу IEC-104 Tools.exe, которую создал нам Баублис Дмитрий, пока еще у нас работал (потом ушел зашибать бабки работая программистом у Бикмуллина).



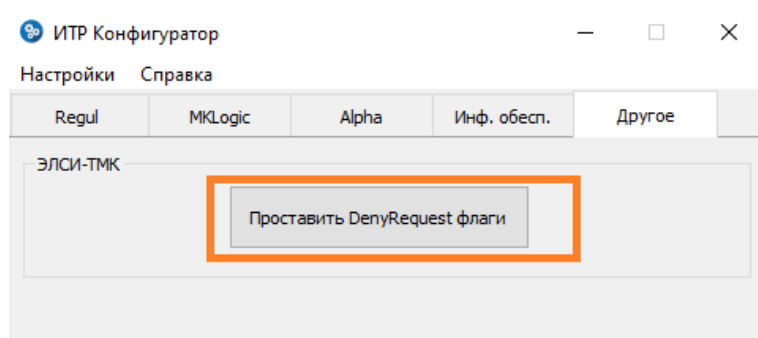
Рисунок

Импортировав файл описаний в программу IEC-104 Tools.exe появляется возможность отслеживать МЭК сигналы более удобно, благодаря подгрузившимся описаниям сигналов.

Вкладка Другое

ЭЛСИ-ТМК

Проставить Deny Request флаги



Рисунок

Данная опция позволяет загрузить один или несколько файлов крейта ЭЛСИ-ТМК.xml, экспортируемых программой ACS Конфигуратор.

И исправить флаги DenyRequest во всех найденных модулях TN713 (то есть взвести их в true).

Потому что несмотря на то, что на всех modbus slave устройствах в канале предоставлено одно и то же значение «Запрет выдачи опроса», программа ACS Configurator может случайно игнорировать эту настройку.

Модифицированный файл крейта сохраняется рядом с загруженным файлом и имеет постфикс _mod.