Резервированные системы на базе контроллеров линейки SM250.

Типы резервирования

В основе систем АСУТП на базе программируемых контроллеров лежат следующие принципы.
Системы включают в себя, как правило два контроллера.
Оба контроллера, имеют одинаковую конфигурацию, одинаковую внутреннюю операционную систему, в них загружены одинаковые прикладные программы. Оба контроллера находятся в рабочем состоянии.
Полный доступ к объектам управления (далее УСО) имеет только один контроллер, являющийся в системе Основным. Он может как читать данные из УСО, так и записывать их.
Второй контроллер является Резервным. Он может только читать данные из УСО. Доступ к записи в УСО для Резервного контроллера закрыт.
Дополнительно может быть реализована пересылка полученных из УСО данных в память резервного контроллера из памяти основного контроллера по каналу синхронизации.
При выходе из строя Основного контроллера, или при возникновении в нём определённых разработчиком ошибок происходит переключение: Резервный контроллер становится Основным. Бывший Основной контроллер или выключается или переводится в состояние, не влияющее на объекты управления, посредством закрытия для него доступа к УСО. Происходит так называемое «свапирование» или ввод резерва. Регулирование доступа к УСО осуществляется на программно-аппаратном уровне.

Системы АСУТП с резервированием можно разделить на следующие группы:
«Hot-Hot». В этих системах оба контроллера выполняют прикладные программы полностью.
«Hot-Standby». В этих системах Основной контроллер выполняет прикладную программу полностью, а Резервный только в части, заданной при конфигурировании.
Важной особенностью этих систем является точная синхронизация начала каждого SCAN-цикла в контроллерах и малое время ввода резерва. Идеальным является время ввода резерва, равное полутора-двум SCAN-циклам контроллера.
«Warm Standby». В этом случае время ввода резерва может быть достаточно большим, до нескольких секунд. Требования к синхронизации SCAN-циклов контроллеров могут не выставляться.

Для построения полнофункциональных систем с резервированием основные производители контроллерного оборудования выпускают специализированные процессорные и сопроцессорные модули с поддерживающими резервирование внутренними операционными системами. Эти модули позволяют организовать резервирование на аппаратном уровне. Но в случае отсутствия у вендора таких модулей можно попробовать построить систему с резервированием на программном уровне.

В лаборатории Центра Компетенций «Систем Электрик» была проведена работа по исследованию возможностей построения систем с резервированием на базе линейки SM250 с процессорными модулями SM252MESC. Ход работ, выводы и комментарии представлены ниже.

Синхронизация

Одной из основных задач при построении резервированных систем на базе PLC является синхронизация работы входящих в систему Основного и Резервного контроллеров.
Для обеспечения устойчивого резервирования нужен несвапируемый канал синхронизации. То есть канал связи, адреса конечных устройств которого не меняются в зависимости от состояния «Основной» или «Резервный», и который обеспечивает обмен данными в обе стороны одновременно.
Канал синхронизации — это высокоскоростной полнодуплексный канал передачи данных (далее синхромассив) между Основным и Резервным контроллером. Этот канал предназначен для следующих операций.
-Синхронизация времени между Основным и Резервным контроллерами. Точное время может быть получено Основным контроллером по протоколу NTP (SNTP) от сервера времени или от SCADA-системы.
-Синхронизация начала и конца SCAN-циклов контроллеров для синхронного выполнения программ.

-Передача из Основного контроллера в Резервный информации, полученной от SCADA/HMI (уставки, команды, задания, и т.д.).
-Передача из основного контроллера в Резервный результатов промежуточных вычислений для их сравнения и/или использования, если это задано разработчиком.
-Передача из Основного контроллера в Резервный данных, полученных из УСО, если это задано разработчиком.
-Обмен диагностическими данными о работоспособности и состоянии контроллеров для принятия решения о вводе резерва.
Примечание. В некоторых системах при наличии резервированных сетей связи с УСО (RSTP, MRP) диагностические данные для принятия решения о вводе резерва передаются по сетям связи с УСО-второй канал диагностики.

В Лаборатории была выполнена проверка возможности синхронизации Основного и Резервного контроллеров с использованием протокола Modbus RTU (физический интерфейс RS-485) на скорости передачи данных 115200bps.
Для работы использовались контроллеры Systeme Electric SM252MESC (внутренняя операционная система V1.04), работающие в среде разработки приложений Codesys V3.5 SP11 Patch6 с использованием функциональных блоков Modbus RTU Master и Modbus RTU Slave, входящих в библиотеку “Systeme Electric Modbus Library V1.6”.
В случае применения ФБ MBUS_INIT, MBUS_SLAVE, MBUS_MSG, MBUS_CTRL порт Modbus RTU может работать или как Мастер, или как Слейв.
При попытке запустить в одном контроллере MBUS_INIT и MBUS_SLAVE вместе с блоками MBUS_MSG и MBUS_CTRL порт работает только как Slave. При попытке программного переключения с Master-а на Slave при помощи входов на функциональных блоках требуется остановка контроллера (режим Stop в Codesys) с повторным запуском контроллера. То есть нет переключения “на лету”. Это штатная работа ФБ, так как Modbus RTU порт не может изменить своё состояние во время выполнения программы. Это делает невозможным безударное переключение контроллеров при вводе резерва.
Заключение: Протокол Modbus RTU не может быть использован для канала синхронизации.

В Лаборатории была выполнена проверка возможности синхронизации Основного и Резервного контроллеров с использованием протокола Modbus TCP на том же оборудовании. Использовались функциональные блоки “MBUS_TCP_REQ” и “MBUS_TCP_SLAVE”.
В отличие от Modbus RTU можно переключать режимы блоков “MBUS_TCP_REQ” (Write/Read) программой на лету.
Находящийся в Основном контроллере блок “MBUS_TCP_REQ” в режиме записи из Основного контроллера должен был передавать данные для синхронизации в Резервный контроллер.
Находящийся в Резервном контроллере блок “MBUS_TCP_REQ” в режиме считывания должен был брать из Основного контроллера информацию, подтверждающую работоспособность Основного контроллера, по анализу которой Резервный контроллер должен был переводится в режим «Основной» (если нужно). См. Рис.1

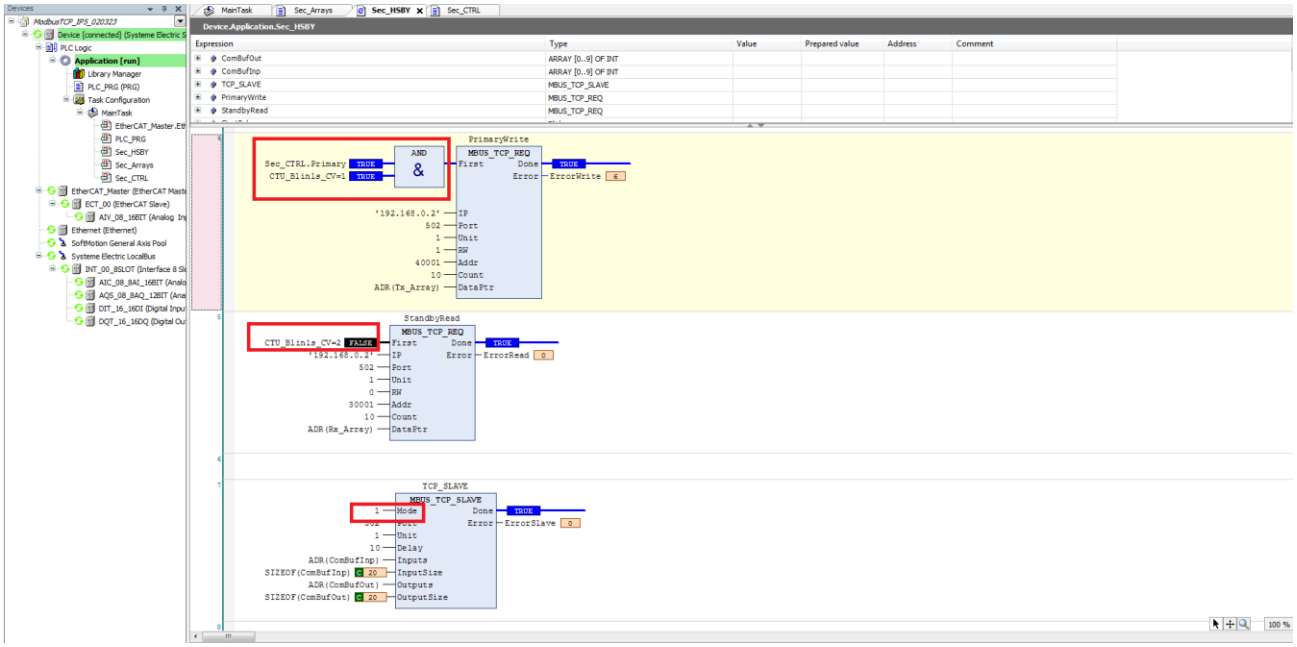


Рис.1

Примечание. Контроллер SM252 может одновременно выполнять только один коммуникационный блок “MBUS_TCP_REQ”. И если размер синхромассива больше 120 слов, то потребуется больше, чем один экземпляр ФБ. И запускать их придётся по очереди. То есть размеры синхромассива влияют на загрузку CPU и SCAN-цикл контроллера.

Во время тестирования смены IP-адресов с помощью ФБ “changeIPAddress”, а также при проработке вариантов диагностики сетей, выяснилось, что коммуникационный блок “MBUS_TCP_REQ” в режиме чтения при потере связи сохраняет предыдущие значения. Нам нужно, чтобы данные канала при обрыве линии или выключении партнёра сбрасывались в ноль. Поэтому этот метод синхронизации не может быть использован для построения системы с резервированием.

Заключение: Протокол Modbus TCP совместно с функциональными блоками “MBUS_TCP_REQ” и “MBUS_TCP_SLAVE” не может быть использован для канала синхронизации.

В Лаборатории проведено тестирование синхронизации с использованием протокола Modbus TCP совместно с сервисом «TCP IO Scanner». Оба контроллера конфигурируются одинаково согласно правилам работы с 3S драйверами Modbus_TCP_Master и Modbus_TCP_Slave. Codesys, см Рис. 2.

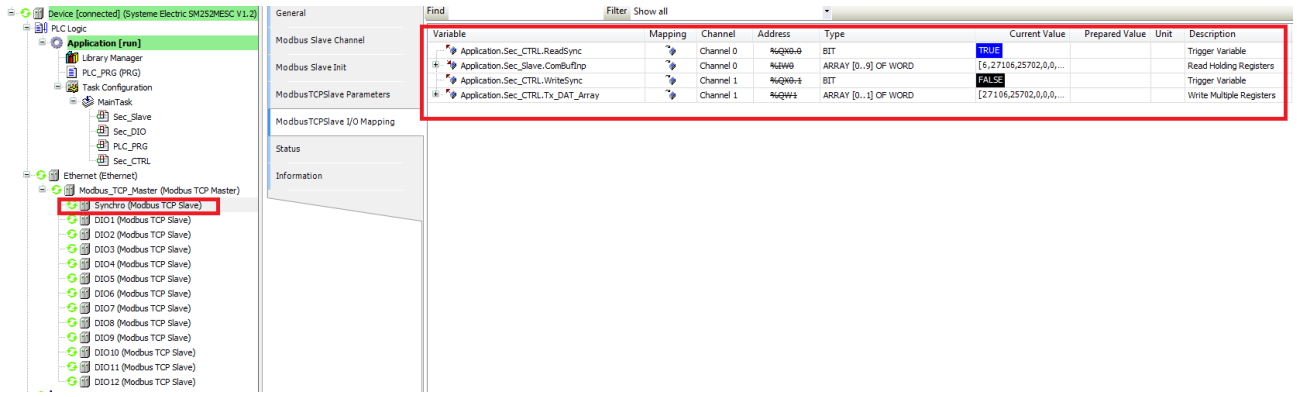


Рис.2

В самом простом варианте настраиваются два канала.

Первый канал (Channel 0 на Рис. 2) предназначен для чтения данных из контроллера-партнёра. Активен в обоих контроллерах. Чтение может активироваться как по переднему фронту бита, интегрирующего необходимые для старта чтения условия, так и циклически, с периодом 100 миллисекунд.
Второй канал (Channel 1 на Рис. 2) предназначен для записи синхромассива из Основного контроллера в Резервный контроллер. Активен в обоих контроллерах. Запись может активироваться как по переднему фронту бита, интегрирующего необходимые для старта записи условия, так и циклически, с периодом 100 миллисекунд.
Примечание. В лаборатории протестированы оба режима записи и чтения (циклический и по событию), оба варианта рабочие.

Если сеть синхронизации активна, на модуле SM3DQ16 программа зажигает светодиод LED8. В случае отсутствия синхронизации этот светодиод гаснет.

Заключение: Вариант с сервисом «TCP IO Scanner» рабочий. Рекомендуется к применению.

Смена IP-адресов.

Если делать комплекс с полноценным резервированием (как Modicon или Simatic), то необходимо обеспечить начальную инициализацию с установкой IP-адресов для каждого контроллера в зависимости от его состояния (Основной или Резервный). И вот здесь начинаются проблемы. Во время начальной инициализации контроллеры «ищут» друг друга в сети и по состоянию партнёра устанавливают собственное состояние. Но иногда теряют партнёра, так как в это же самое время идёт переключение IP-адресов с помощью ФБ “changeIPAddress”. Почему это происходит, установить невозможно, так как при смене IP-адресов программатор Codesys теряет контроллер. Для оценки того, что происходи в контроллере нужен отдельный порт программирования (например USB). Но его нет, и можно о причинах только предполагать. Например, у какой-то момент в сети оказываются два одинаковых IP-адреса или наоборот, в сети не оказывается ни одного адреса, каждый контроллер считает, что он один в сети и становится Primary. И в результате мы получаем два главных контроллера и ни одного резервного. Это происходит из-за отсутствия синхронизации циклов контроллеров, но синхронизировать контроллеры по времени невозможно по причине отсутствия поддержки протокола NTP (SNTP).
Так же трудность представляет тот факт, что после смены IP-адреса контроллера остаются неизменёнными настройки каналов “Modbus_TCP_Slave TCP”, включая и IP-адреса слейв-устройств. Например, контроллер, который до переключения имел IP-адрес 192.168.0.5 (Резервный), читал информацию о статусе партнёра из IP-адреса 192.168.0.2 (Основной). После смены IP-адреса он должен принять IP-адрес 192.168.0.2 (Основной) и читать данные из самого себя. Некорректно. А менять программно настройки каналов “Modbus_TCP_Slave TCP” мы не можем.
Примечание: как было отмечено выше, в настоящих Hot Standby системах синхронизация осуществляется через отдельные коммуникационные сопроцессоры на уровне FW. Это полнодуплексные каналообразующие устройства, IP-адреса которых не меняются в зависимости от состояния Основной-Резервный, а жёстко привязаны к контроллеру «А» или «В». Обмен данными через них занимает часть полосы в шине УСО. В нашем случае сопроцессоров синхронизации нет и не предвидится. Все коммуникации осуществляются на уровне прикладных программ, нагружая и процессор и единственный Ethernet-порт.
Дополнительно в процессе работы имел место случай некорректной работы функционального блока “changeIPAddress”. Похоже, ФБ “changeIPAddress” не доработан до конца.
Заключение: Работа над вариантом со сменой IP-адресов остановлена.

DIO

Планировалось провести оценку использования для сетей DIO следующих протоколов:
-EtherCAT,
-Modbus RTU,
-Modbus TCP с использованием коммуникационных блоков,
-Modbus TCP с использованием коммуникационных сервисов Modbus_TCP_Master и Modbus_TCP_Slave.

Применение EtherCAT невозможно, потому что в случае его использования невозможно организовать программное управление доступом к сети DIO.
Применение Modbus RTU невозможно, потому работающие в режиме чтения функциональные блоки MBUS_MSG в случае обрыва коммуникаций возвращают последнее полученное значение. Нам для диагностики сети необходимо, чтобы была возможность выбора между сохранением последнего значения и установкой полученных данных в ноль.
Применение Modbus TCP с использованием коммуникационных блоков имеет ряд ограничений.
У блока “MBUS_TCP_REQ” есть недостаток, когда он в режиме считывания при обрыве линии сохраняет последние значения, а не сбрасывает их в ноль. Для купирования этой проблемы можно данные от DIO записывать в промежуточный буфер, и при обнаружении ошибки этот буфер обнулять. Тестирование этого варианта было выполнено на стенде, см. ниже, но в качестве рабочего он принят не был.
В качестве рабочего принят протокол Modbus TCP с использованием коммуникационных сервисов Modbus_TCP_Master и Modbus_TCP_Slave.

Тестировались два варианта связи с DIO по сети Ethernet с использованием протокола Modbus TCP и коммуникационных сервисов Modbus_TCP_Master и Modbus_TCP_Slave.
Первый вариант: Основной контроллер и пишет данные в DIO-устройство, и читает данные из него. Резервный контроллер к сети DIO доступа не имеет, а получает данные о состоянии DIO от Основного контроллера по каналу синхронизации.
Второй вариант: Основной контроллер и пишет данные в DIO-устройство, и читает данные из него. Резервный контроллер только читает данные из DIO-устройства. В этом случае значительно снижается нагрузка на канал синхронизации.
Примечание. Данные варианты коммуникаций могут быть использованы и для межконтроллерного обмена данными.

На стенде тестировался второй вариант, см. Рис 3-а.

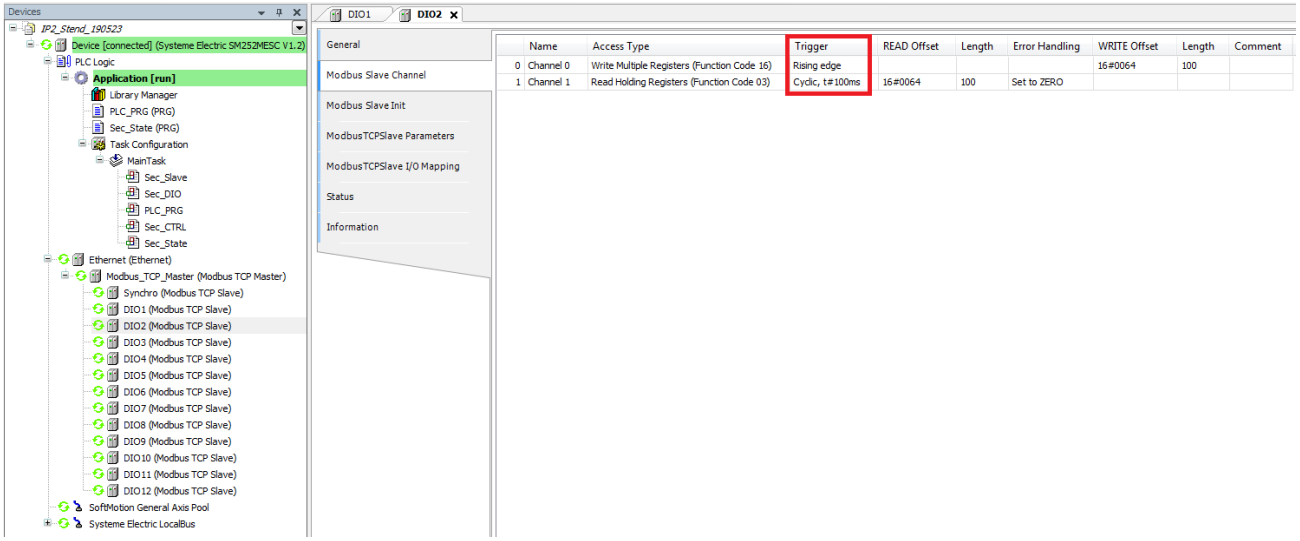


Рис. 3-а

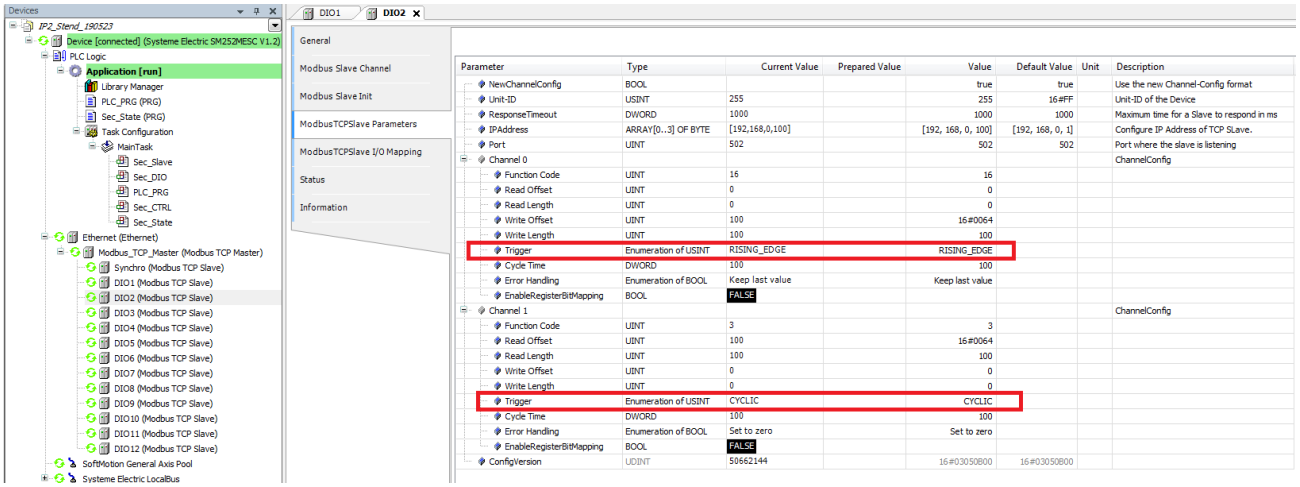


Рис. 3-б

Особенностью этой конфигурации является то, что запись в DIO-устройство ведётся по фронту команды на запись (WriteDIO), см. Рис. 3-б.
Чтение из DIO ведётся постоянно, обоими контроллерами, и Основным, и Резервным, см. Рис.3-в.

```
//Импульсы циклов записи в DIO. Период подбирается при отладке программы или синхронизируется со скан-циклом контроллера
BlinkDIOWrite(ENABLE:=NOT Sec_CTRL.INIT, TIMELOW:=T#50MS, TIMEHIGH:=T#50MS);
IF Sec_CTRL.ThisPrimary THEN WriteDIO:=BlinkDIOWrite.OUT; ELSE WriteDIO:=FALSE; END_IF;
// Измерение частоты запросов записи в DIO
DIO_FREQ_MEASURE(IN:=WriteDIO, PERIODS:=10, OUT=>FreqWriteDIO);
```

Рис. 3-в

На стенде тестировался обмен данными с двенадцатью DIO-устройствами через сканнер (Modbus TCP Master/Modbus TCP Slave сервис. DIO-устройства собраны на базе контроллера M340. В каждое удалённое устройство основной контроллер записывал данные (по 100 Modbus-регистров) и считывал эти же регистры. Резервный контроллер считывал эти же данные из DIO устройств.
На полный цикл опроса (чтение и запись) всех десяти удалённых устройств потребовалось не более 150 миллисекунд. Для этой платформы это очень хороший результат.

Для платформы SM250 можно сконфигурировать до 32-х DIO устройств (Modbus TCP Slave). Так, когда среда Codesys позволяет добавить устройство “Modbus TCP Slave”, то доступна опция “Append Device”, см. Рис. 4.

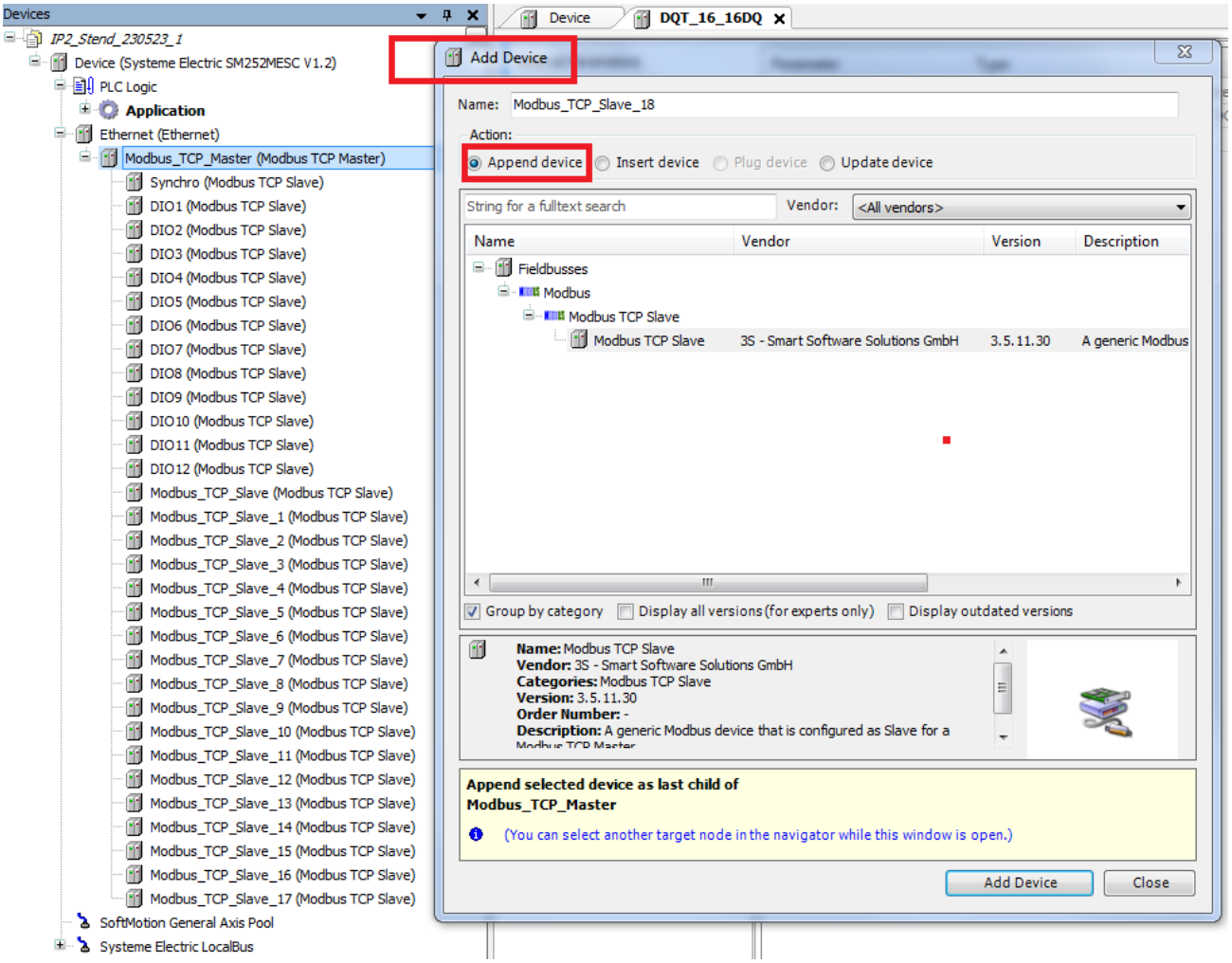


Рис.4

В случае попытки сконфигурировать более 32-х устройств “Modbus TCP Slave” опция “Append Device” заблокирована в Codesys, см. Рис. 5-а.

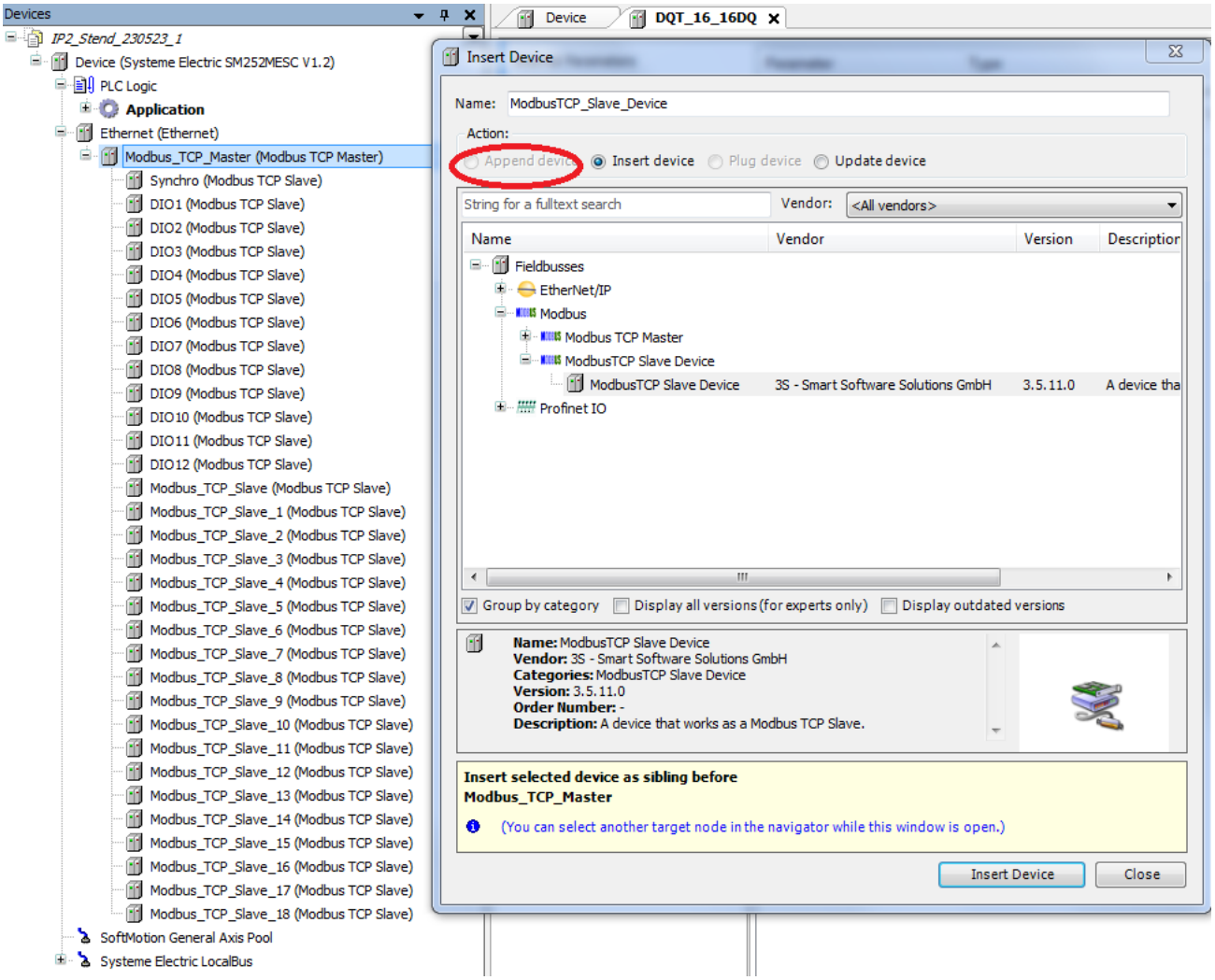


Рис.5-а

Реального влияния количества сконфигурированных устройств Modbus_TCP_Slave на общее время опроса всех DIO-устройств нашими средствами зафиксировано не было.

При увеличении количества сконфигурированных каналов внутри устройства Modbus_TCP_Slave наблюдалась следующая картина.
Среда разработки приложений Codesys для данной платформы не выдает сообщений о допустимом количестве каналов внутри устройства Modbus_TCP_Slave. Так на стенде было сконфигурирован 41 канал внутри устройства DIO1. Приложение загрузилось в контроллер и работало без аварийных сообщений или предупреждений. Но количество каналов оказывает заметное влияние на общее время опроса всех DIO-устройств. Так при периоде сканирования каждого канала, равной 100 миллисекундам общее время опроса примерно равно количеству каналов, умноженному на половину периода сканирования, см. Рис. 5-б.
Примечание. Данная платформа предназначена для работы с УСО небольшой с информационной ёмкостью, например с островами Wellink. Поэтому использование большого количества каналов внутри DIO-устройства не планируется, и указанная выше особенность не является критичной.

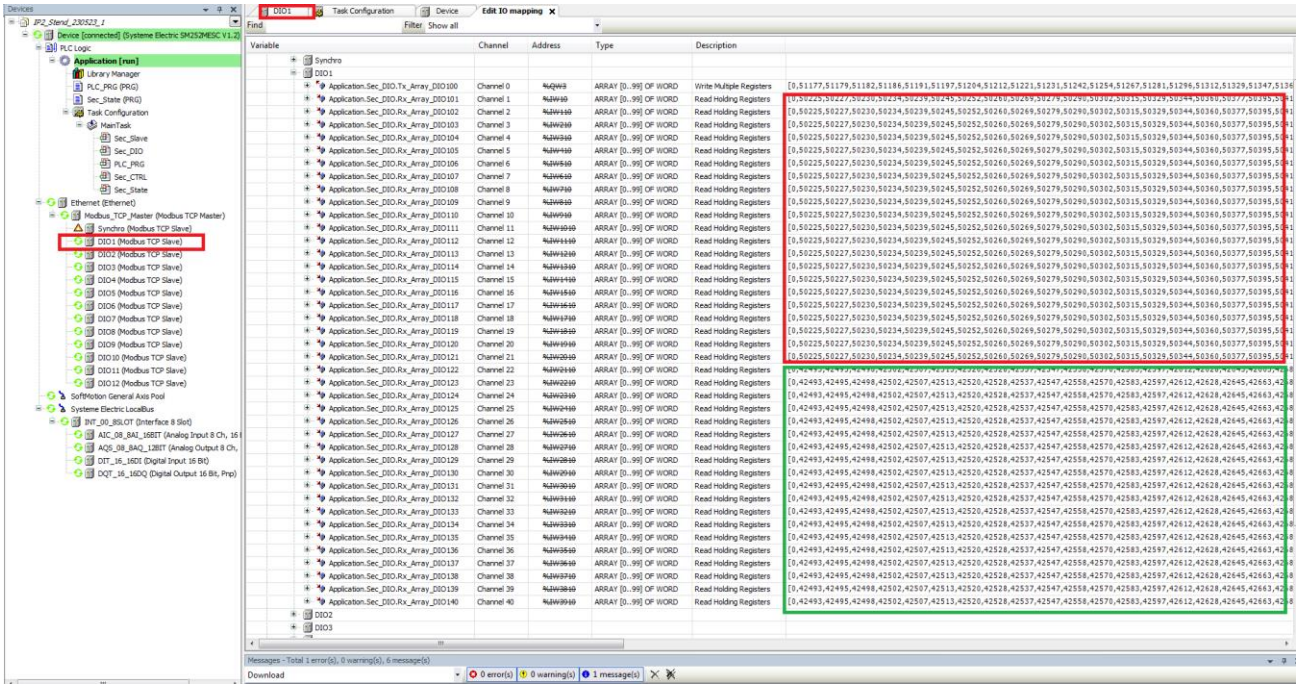


Рис. 5-б

Что касается диагностики каналов, то в работе коммуникационных сервисов Modbus_TCP_Master и Modbus_TCP_Slave на уровне системных библиотек Codesys возможна аппаратная диагностика только Slave-устройства целиком. Диагностика каналов внутри устройства не предусмотрена, а их может быть минимум два (канал чтения и канал записи). Для решение этой проблемы мы используем программную диагностику, которая будет рассмотрена ниже.
Примечание. Полноценного аналога Ethernet I/O Scanner, какой был у Modicon, не получилось, подвела аппаратная диагностика, есть только программная.

На стенде был проверен вариант использования для обмена данными с DIO устройствами последовательно, друг после друга, запускаемых блоков “MBUS_TCP_REQ”.
Проведено тестирование с имитацией DIO-устройств на одном контроллере M340. К контроллеру M340 обращаются в режиме чтения данных десять экземпляров блоков “MBUS_TCP_REQ”. Каждый выполнивший запрос блок запускает следующий за на ним блок. В качестве стартового импульса используются импульсы с разными периодами, смотрите Рис.6-Рис.10.

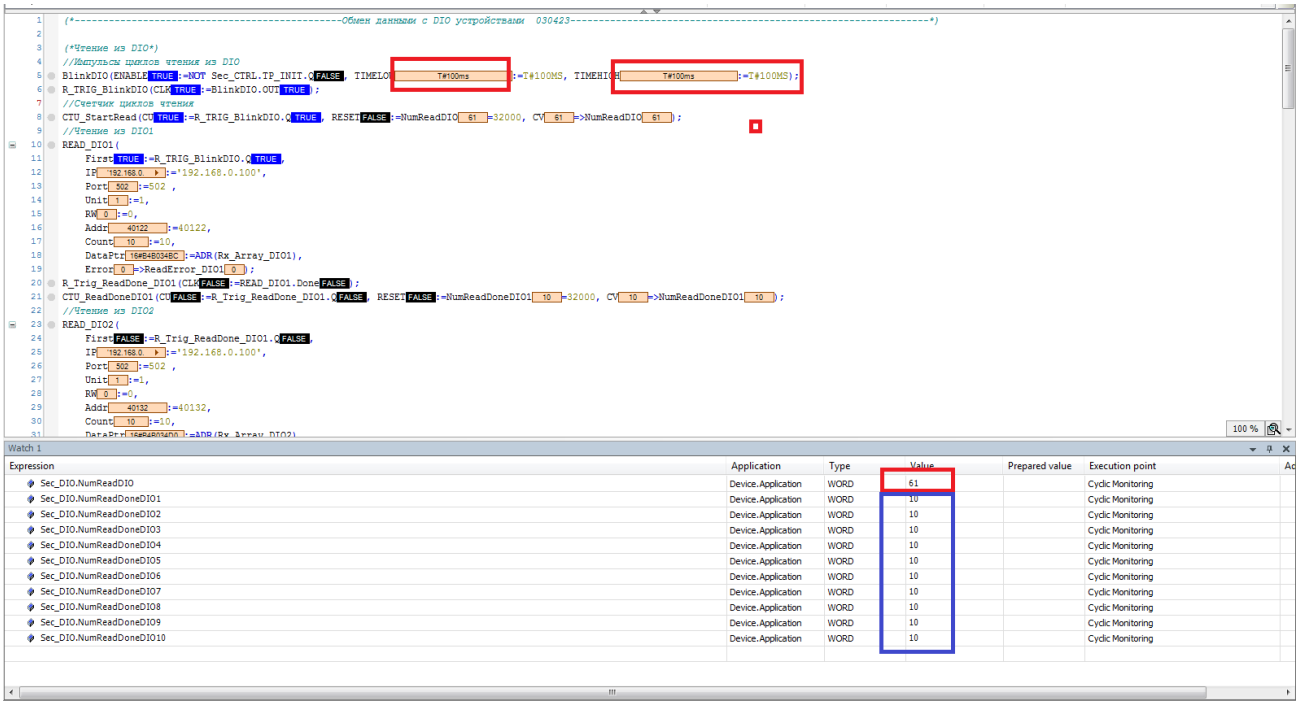

(*----------Обмен данными с DIO устройствами 030423----------*)
(*Чтение из DIO*)
//Импульсы циклов чтения из DIO
BlinkDIO(ENABLE:=NOT Sec_CTRL.TP_INIT.Q, TIMELOW:=T#100MS, TIMEHIGH:=T#100MS);
R_TRIG_BlinkDIO(CLK:=BlinkDIO.OUT);
//Счетчик циклов чтения
CTU_StartRead(CU:=R_TRIG_BlinkDIO.Q, RESET:=NumReadDIO=32000, CV=>NumReadDIO);
//Чтение из DIO1
READ_DIO1(
First:=R_TRIG_BlinkDIO.Q,
IP:='192.168.0.100',
Port:=502 ,
Unit:=1,
RW:=0,
Addr:=40122,
Count:=10,
DataPtr:=ADR(Rx_Array_DIO1),
Error=>ReadError_DIO1);
R_Trig_ReadDone_DIO1(CLK:=READ_DIO1.Done);
CTU_ReadDoneDIO1(CU:=R_Trig_ReadDone_DIO1.Q, RESET:=NumReadDoneDIO1=32000, CV=>NumReadDoneDIO1);
//Чтение из DIO2
READ_DIO2(
First:=R_Trig_ReadDone_DIO1.Q,
IP:='192.168.0.100',
Port:=502 ,
Unit:=1,
RW:=0,
Addr:=40132,
Count:=10,
100 %
Watch 1
Expression
Application
Type
Value
Prepared value
Execution point
Sec_DIO.NumReadDIO
Device.Application
WORD
61
Cyclic Monitoring
Sec_DIO.NumReadDoneDIO1
Sec_DIO.NumReadDoneDIO2
Sec_DIO.NumReadDoneDIO3
Sec_DIO.NumReadDoneDIO4
Sec_DIO.NumReadDoneDIO5
Sec_DIO.NumReadDoneDIO6
Sec_DIO.NumReadDoneDIO7
Sec_DIO.NumReadDoneDIO8
Sec_DIO.NumReadDoneDIO9
Sec_DIO.NumReadDoneDIO10
10

Рис. 6

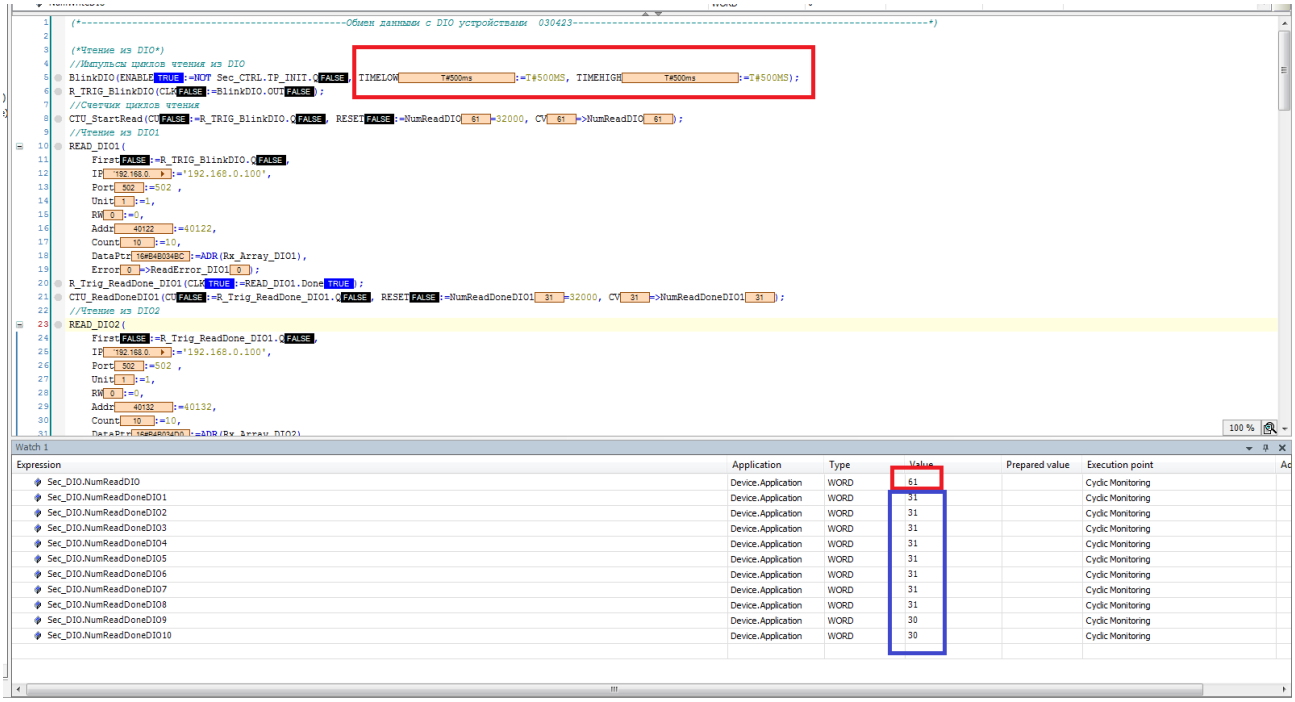
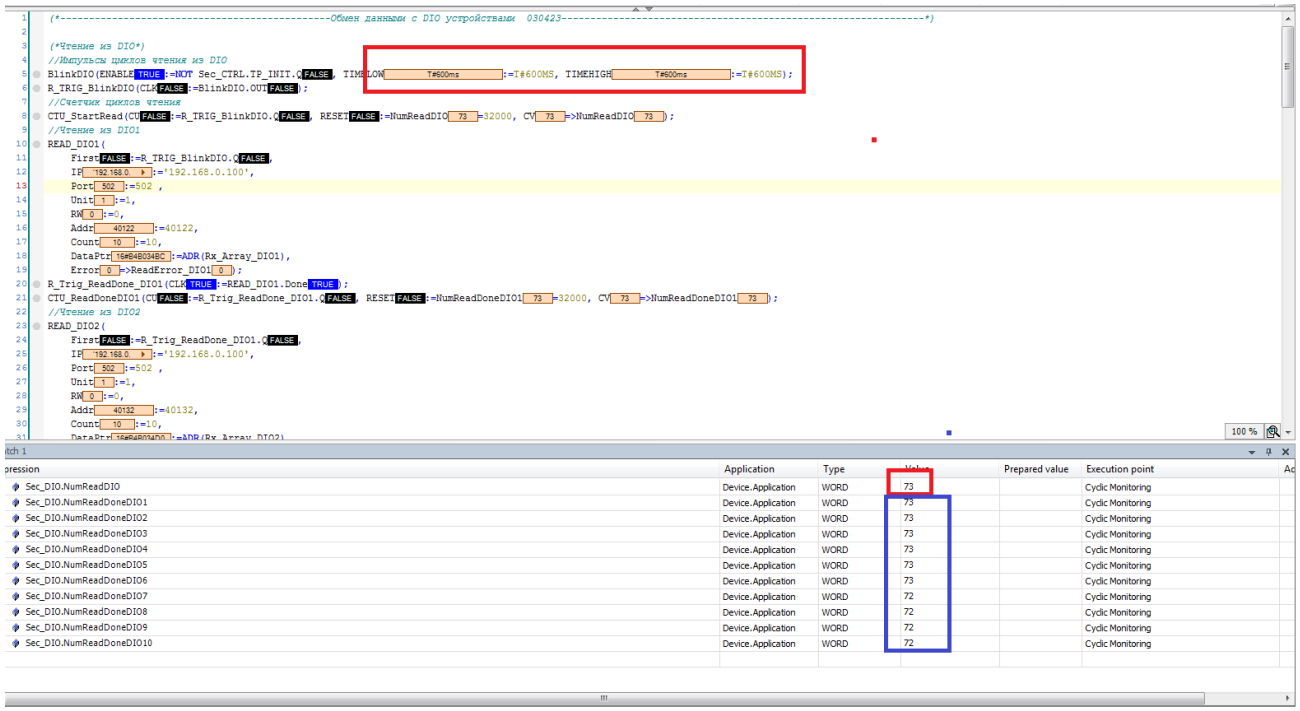

(*----------Обмен данными с DIO устройствами 030423----------*)
(*Чтение из DIO*)
//Импульсы циклов чтения из DIO
BlinkDIO(ENABLE:=NOT Sec_CTRL.TP_INIT.Q, TIMELOW:=T#500MS, TIMEHIGH:=T#500MS);
R_TRIG_BlinkDIO(CLK:=BlinkDIO.OUT);
//Счетчик циклов чтения
CTU_StartRead(CU:=R_TRIG_BlinkDIO.Q, RESET:=NumReadDIO=32000, CV=>NumReadDIO);
//Чтение из DIO1
READ_DIO1(
First:=R_TRIG_BlinkDIO.Q,
IP:='192.168.0.100',
Port:=502 ,
Unit:=1,
RW:=0,
Addr:=40122,
Count:=10,
DataPtr:=ADR(Rx_Array_DIO1),
Error=>ReadError_DIO1);
R_Trig_ReadDone_DIO1(CLK:=READ_DIO1.Done);
CTU_ReadDoneDIO1(CU:=R_Trig_ReadDone_DIO1.Q, RESET:=NumReadDoneDIO1=32000, CV=>NumReadDoneDIO1);
//Чтение из DIO2
READ_DIO2(
First:=R_Trig_ReadDone_DIO1.Q,
IP:='192.168.0.100',
Port:=502 ,
Unit:=1,
RW:=0,
Addr:=40132,
Count:=10,
100 %
Watch 1
Expression
Application
Type
Value
Prepared value
Execution point
Sec_DIO.NumReadDIO
Device.Application
WORD
61
Cyclic Monitoring
Sec_DIO.NumReadDoneDIO1
Sec_DIO.NumReadDoneDIO2
Sec_DIO.NumReadDoneDIO3
Sec_DIO.NumReadDoneDIO4
Sec_DIO.NumReadDoneDIO5
Sec_DIO.NumReadDoneDIO6
Sec_DIO.NumReadDoneDIO7
Sec_DIO.NumReadDoneDIO8
Sec_DIO.NumReadDoneDIO9
Sec_DIO.NumReadDoneDIO10
31
30

Рис.7

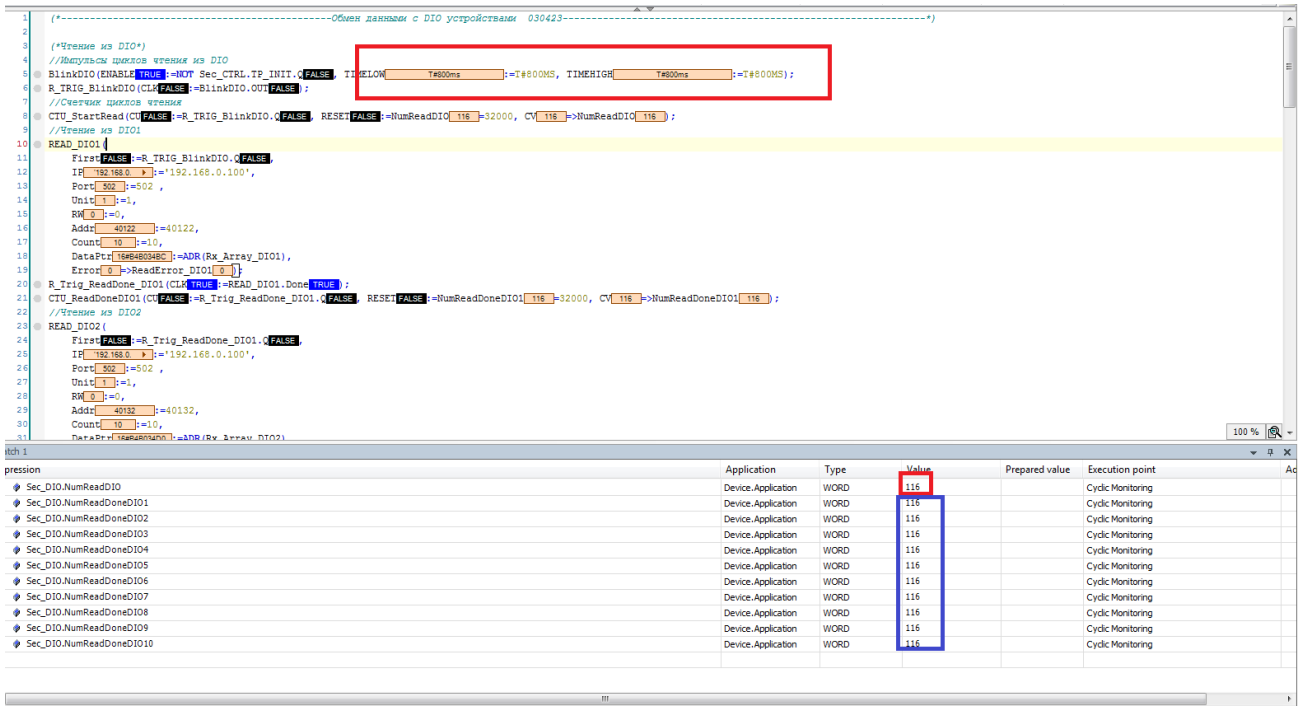


```
(*-----------------------------------------Обмен данными с DIO устройствами  030423-----------------------------------------*)

(*Чтение из DIO*)
//Импульсы циклов чтения из DIO
BlinkDIO(ENABLE:=NOT Sec_CTRL.TP_INIT.Q, TIMELOW:=T#600MS, TIMEHIGH:=T#600MS);
R_TRIG_BlinkDIO(CLK:=BlinkDIO.OUT);
//Счетчик циклов чтения
CTU_StartRead(CU:=R_TRIG_BlinkDIO.Q, RESET:=NumReadDIO=32000, CV=>NumReadDIO);
//Чтение из DIO1
READ_DIO1(
    First:=R_TRIG_BlinkDIO.Q,
    IP:='192.168.0.100',
    Port:=502 ,
    Unit:=1,
    RW:=0,
    Addr:=40122,
    Count:=10,
    DataPtr:=ADR(Rx_Array_DIO1),
    Error=>ReadError_DIO1);
R_Trig_ReadDone_DIO1(CLK:=READ_DIO1.Done);
CTU_ReadDoneDIO1(CU:=R_Trig_ReadDone_DIO1.Q, RESET:=NumReadDoneDIO1=32000, CV=>NumReadDoneDIO1);
//Чтение из DIO2
READ_DIO2(
    First:=R_Trig_ReadDone_DIO1.Q,
    IP:='192.168.0.100',
    Port:=502 ,
    Unit:=1,
    RW:=0,
    Addr:=40132,
    Count:=10,
```

pression	Application	Type	Value	Prepared value	Execution point	Ac
Sec_DIO.NumReadDIO	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO1	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO2	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO3	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO4	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO5	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO6	Device.Application	WORD	73		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO7	Device.Application	WORD	72		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO8	Device.Application	WORD	72		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO9	Device.Application	WORD	72		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO10	Device.Application	WORD	72		Cyclic Monitoring	

Рис.8



```
(*-----------------------------------------Обмен данными с DIO устройствами  030423-----------------------------------------*)

(*Чтение из DIO*)
//Импульсы циклов чтения из DIO
BlinkDIO(ENABLE:=NOT Sec_CTRL.TP_INIT.Q, TIMELOW:=T#800MS, TIMEHIGH:=T#800MS);
R_TRIG_BlinkDIO(CLK:=BlinkDIO.OUT);
//Счетчик циклов чтения
CTU_StartRead(CU:=R_TRIG_BlinkDIO.Q, RESET:=NumReadDIO=32000, CV=>NumReadDIO);
//Чтение из DIO1
READ_DIO1(
    First:=R_TRIG_BlinkDIO.Q,
    IP:='192.168.0.100',
    Port:=502 ,
    Unit:=1,
    RW:=0,
    Addr:=40122,
    Count:=10,
    DataPtr:=ADR(Rx_Array_DIO1),
    Error=>ReadError_DIO1);
R_Trig_ReadDone_DIO1(CLK:=READ_DIO1.Done);
CTU_ReadDoneDIO1(CU:=R_Trig_ReadDone_DIO1.Q, RESET:=NumReadDoneDIO1=32000, CV=>NumReadDoneDIO1);
//Чтение из DIO2
READ_DIO2(
    First:=R_Trig_ReadDone_DIO1.Q,
    IP:='192.168.0.100',
    Port:=502 ,
    Unit:=1,
    RW:=0,
    Addr:=40132,
    Count:=10,
```

pression	Application	Type	Value	Prepared value	Execution point	Ac
Sec_DIO.NumReadDIO	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO1	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO2	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO3	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO4	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO5	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO6	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO7	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO8	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO9	Device.Application	WORD	116		Cyclic Monitoring	
Sec_DIO.NumReadDoneDIO10	Device.Application	WORD	116		Cyclic Monitoring	

Рис.9

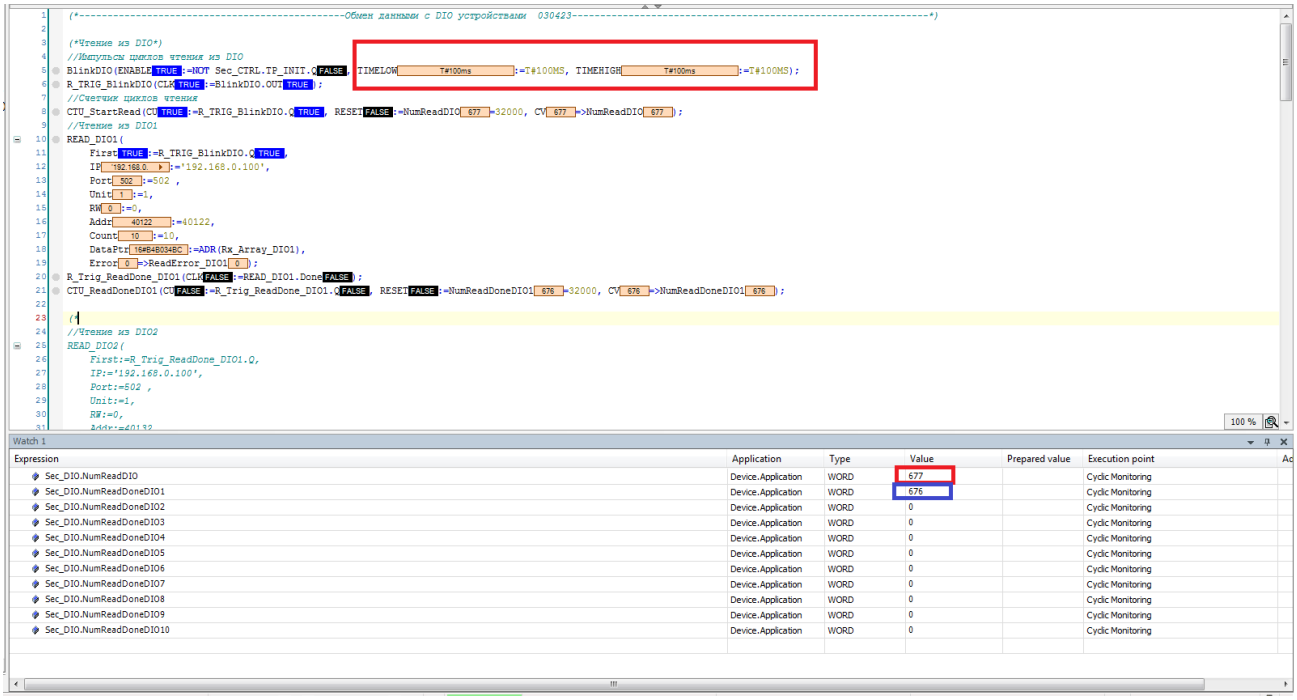


Рис. 10

Было протестировано обращение к имитации 10-ти DIO-устройств, десять запросов на чтение, десять запросов на запись, итого последовательно, один за другим запускалось 20 экземпляров блока “MBUS_TCP_REQ”.
Особенностью функциональных блоков “MBUS_TCP_REQ” является тот факт, что одновременно (в данный момент времени) может выполняться только один блок. Запускать выполнение блоков строго по одному, друг за другом, что и было реализовано на стенде. Поэтому общее время опроса DIO-сети равняется произведению времени опроса одного DIO-устройства на количество устройств.
Поэтому этот вариант нас не может удовлетворить. Для нормальной работы контроллеров в реальном масштабе времени необходимо, чтобы устройства ввода-вывода опрашивались синхронно (или близко к этому) со скан-циклом контроллера. На каждый скан контроллера должен приходиться один цикл опроса всех DIO-устройств в системе.
Заключение. Для обмена данными с удалёнными устройствами предпочтительней использование коммуникационных сервисов Modbus_TCP_Master и Modbus_TCP_Slave.

Светодиодная индикация.

Для контроля состояния контроллеров и отслеживания переключений режимов на светодиодную индикацию модулей SM3DQ16 выведена следующая информация:
-Этот контроллер Primary: LED0
-Этот контроллер Standby: LED1
-Этот контроллер резервирован: LED2
-Этот контроллер Offline: LED3
-Выполняется начальная инициализация: LED4
-Удалённый контроллер Primary: LED6
-Удалённый контроллер Standby: LED7
-Сеть синхронизации в работе: LED8
-Сеть DIO в работе: LED9

Начальная инициализация и нормальная работа.

Рассмотрим режим на примере контроллера «А». Для управления начальной инициализацией и вводом резерва используется разработанный нами функциональный блок “CTRL”, код блока приведен ниже, см. Рис.11. Блок вызывается в программной секции “Sec_CTRL”, см. Рис.12.

```
FUNCTION_BLOCK CTRL
(*-------------------------------------------------------------Управление резервированием 130423--OR-------------------------------------------------------------------------*)

(*Начальная инициализация*)
TP_INIT(IN:=TRUE , PT:=PulseINIT);
IF TP_INIT.Q THEN ThisPRM:=FALSE; ThisSTB:=FALSE; END_IF;
IF TP_INIT.Q THEN INIT:=TRUE; ELSE INIT:=FALSE; END_IF;

(*Анализ состояния удалённого контроллера по данным из канала синхронизации или из модуля DI*)
IF NetRemPRM OR DiRemPRM THEN RemPRM:=TRUE; ELSE RemPRM:=FALSE; END_IF;
IF NetRemSTB OR DiRemSTB THEN RemSTB:=TRUE; ELSE RemSTB:=FALSE; END_IF;
IF NetRemRes OR DiRemRES THEN RemRES:=TRUE; ELSE RemRES:=FALSE; END_IF;

(*Рабочие режимы*)
IF NOT TP_INIT.Q THEN
   //Условия ввода основного режима (Primary).
   IF (RemSTB OR NOT RemRES) AND NOT NetError THEN ThisPRM:=TRUE; ELSE ThisPRM:=FALSE; END_IF;
   //Условия ввода резервного режима контроллера (Standby)
   IF RemPRM AND NOT NetError THEN ThisSTB:=TRUE; ELSE ThisSTB:=FALSE; END_IF;
   //Есть резервирование
   IF ThisSTB OR ThisPRM THEN ThisRES:=TRUE; ELSE ThisRES:=FALSE; END_IF;
    //Режим Offline
   IF NetError THEN ThisSTB:=FALSE; ThisSTB:=FALSE; END_IF;
   IF NetError OR NOT ThisRES THEN ThisOff:=TRUE; ELSE ThisOff:=FALSE; END_IF;
END_IF;
```

Рис.11

После включения контроллера «А» по переднему фронту стартового бита “TRUE” запускается импульс начальной инициализации длительностью, заданной в параметре “Pulse_Init” (10 секунд для стенда). При этом на DQ зажигается LED4. За время импульса инициализации успевают запуститься сети Ethernet, и контроллер начинает читать по синхроканалу состояния контроллера «В».

После окончания начальной инициализации (установки в ноль импульса “TP_INIT.Q”, если контроллера «В» в сети нет, то контроллер «А» становится Основным (Primary) при выполнении условий перехода в состояние “Primary”:
-контроллер исправен;
-есть связь с DIO устройствами по слейв-каналам;
-нет блокировки от программы или внешних сигналов на установку режима Primary.
При этом на DQ зажигается LED0.
Если эти условия не соблюдены, то контроллер остаётся в режиме “Offline“.

Если после окончания начальной инициализации и выполнения стартовых условий в сети уже есть контроллера «В», и он Основной (имеет состояние Primary), то контроллер «А» устанавливается в режим Резервный (Standby), на его модуле DQ зажигаются LED1 и LED6. Если состояние контроллера «В» Резервный (Standby), то контроллер «А» становится Primary, на DQ зажигаются LED0 и LED7.

Поведение контроллера «В» при начальной инициализации аналогично поведению контроллера «А».

Ввод резерва (свапирование).

Управление вводом резерва, (свапированием, переходом контроллера из состояние Резервный (Standby) в состояние Основной (Primary)), также осуществляется с помощью функционального блока “CTRL” (Рис.11), который вызывается в программной секции “Sec_CTRL”, см. Рис.12.
Если Основной контролер (Primary) перестаёт работать, то вместо него основным контроллером становится бывший Резервный (Standby). Неисправный контроллер Primary отключается от DIO- сети и принимает состояние в зависимости от причины невозможности быть Основным.
Контроллер перестаёт быть Основным в следующих случаях:
-при выключении,
-при выходе из строя,
-при переводе его в состояние Standby программно или по внешнему сигналу,
-при потере связи с DIO-устройствами.

Резервный контроллер получает информацию о том, что Основной контроллер перестал выполнять функции Основного через канал синхронизации или через дискретные входы.

Если Резервный контроллер (Standby) удовлетворяет условиям перехода в состояние Основной (Primary), то он становится Основным. На модулях DQ обоих контроллеров зажигается соответствующая индикация. Резервный контроллер переходит в режим Основного по одному из следующих условий.
-Получение от партнёра по резервированию информации, что он перестал быть Основным по синхроканалу. Для этого входу NetRemPRM экземпляра “CTRL_Swap” блока “CTRL” присваивается полученный через синхроканал ноль.
- Получение от партнёра по резервированию информации, что он перестал быть Основным через дискретный вход. Для этого входу DiRemPRM экземпляра “CTRL_Swap” блока “CTRL” присваивается полученный через дискретный вход ноль.

```
(*----------------------------Управление резервированием 140423,OR, Это контроллер "A". IP=192.168.0.2:502-

(*Управление свапированием*)
    CTRL_Swap(
    PulseINIT:=T#10S,
    PulseSYNC:=T#50MS,
    NetRemPRM:=Sec_Slave.ComBufInp[0].0,
    NetRemSTB:=Sec_Slave.ComBufInp[0].1,
    NetRemRes:=Sec_Slave.ComBufInp[0].2,
    DiRemPRM:=DI_RemPrimary,
    DiRemSTB:=DI_RemStandby,
    DiRemRes:=DI_RemReserved,
    NetError:=Sec_DIO.NetError,
    INIT=>INIT,
    ThisPRM=>ThisPrimary,
    ThisSTB=>ThisStandby,
    ThisRES=>ThisReserved,
    ThisOff=>ThisOffline,
    RemPRM=>RemPrimary,
    RemSTB=>RemStandby,
    RemRES=>RemReserved,
    ReadSync=>ReadSync,
    WriteSync=>WriteSync);
```

Рис.12

При потере канала синхронизации тот контроллер, который сохранил связь с DIO, становится Primary. Потерявший связь с DIO устройствами контроллер переходит в режим Offline. Для возвращения контроллера в рабочие режимы необходимо через Codesys войти в программу, сбросить командой триггер ошибок сети и счётчик ошибок сети. Об этом подробно будет указано ниже. После этого контроллер автоматически примет рабочее состояние (Primary или Standby) в зависимости от состояния контроллера-партнёра.

Необходимо иметь ввиду следующую особенность аппаратной платформы SM250. При переводе контроллера в состояние “Stop” из среды Codesys или при переходе контроллера в это состояние при загрузке прикладных программ светодиодная индикация на лицевой панели модуля SM3DQ16 не меняется и не отражает реального состояния контроллера. Также в этом случае не отражают реального состояния контроллера и значения битов состояния резерва ThisPrimary, ThisStandby, ThisReserved.
Причина в том, что при переходе контроллера в состояние “Stop” невозможно установить биты состояния резерва в неактивное состояние.
Системные средства среды Codesys позволяют определять внутри программы текущее состояние контроллера (Run или Stop), для этого можно использовать указатель на состояния приложения см. Рис. 13.

```
PROGRAM POU
VAR
    p1: POINTER TO CmpApp.RTS_IEC_RESULT;
    p2: POINTER TO CmpApp.APPLICATION;
    Name: STRING:='Application';
    R_Trig_Run: R_TRIG;
    PLC_Run: INT;
    R_Trig_Stop: R_TRIG;
    PLC_Stop: INT;
END_VAR;
```

```
(*-----------------------------------------------------------------190523-----------------------------------------------------------------*)

p2:=AppFindApplicationByName(pszString:=Name, pResult:=p1);

R_Trig_Run(CLK:=p2^.udiState=1); //Указатель равен "1", если контроллер в состоянии Run
IF R_Trig_Run.Q THEN
    PLC_Run:=PLC_Run+1;
END_IF;

R_Trig_Stop(CLK:=p2^.udiState=2); //Указатель равен "2", если контроллер в состоянии Stop
IF R_Trig_Stop.Q THEN
    PLC_Stop:=PLC_Stop+1;
END_IF;



(*-----------------------------------------------------------------Конец секции-----------------------------------------------------------------*)
```

Рис. 13

Но при переходе контроллера в состояние “Stop” выполнение программы прекращается, контроллер успевает только установить указатель в соответствующее останову значение (два). Дальнейшая обработка значений указателя для перевода битов состояния резерва в неактивное состояние невозможна, так как контроллер программу не выполняет. Получается некорректная ситуация. Бывший Основной контроллер программу не выполняет, к сети DIO доступа не имеет, бывший Резервный контроллер стал Основным, управление процессом не прерывается. Но на внешней индикации и в таблице тегов для SCADA/HMI находящийся в состоянии “Stop” контроллер по-прежнему является Основным.
Эту проблему можно устранить, если для SCADA/HMI использовать не переменные ThisPrimary, ThisStandby, а значение указателя на состояние приложения, см. Рис. 14-а

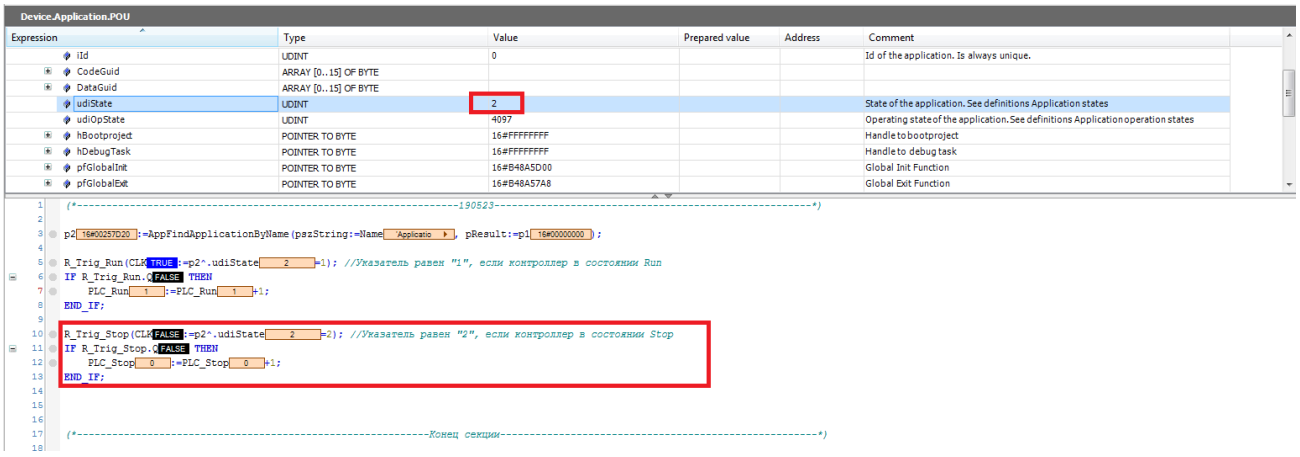


Рис. 14-а

На Рис. 14-а видно, что указатель на состояние приложения имеет значение «два», что соответствует состоянию “Stop” контроллера. Но данное действие возможно только при работе через OPC сервер,

работа через прямые драйвера невозможна, так как не будет выполняться ввод изменения состояния в соответствующие Modbus – регистры.

С точки зрения неадекватной индикации на лицевой панели модуля SM3DQ16 то эту проблему можно решить следующим образом. В настройках контроллера необходимо установить условие обновления выходов таким образом, чтобы в случае перехода контроллера в состояние “Stop” выходы принимали состояние «по умолчанию (Set all output to default), см. Рис. 14-б

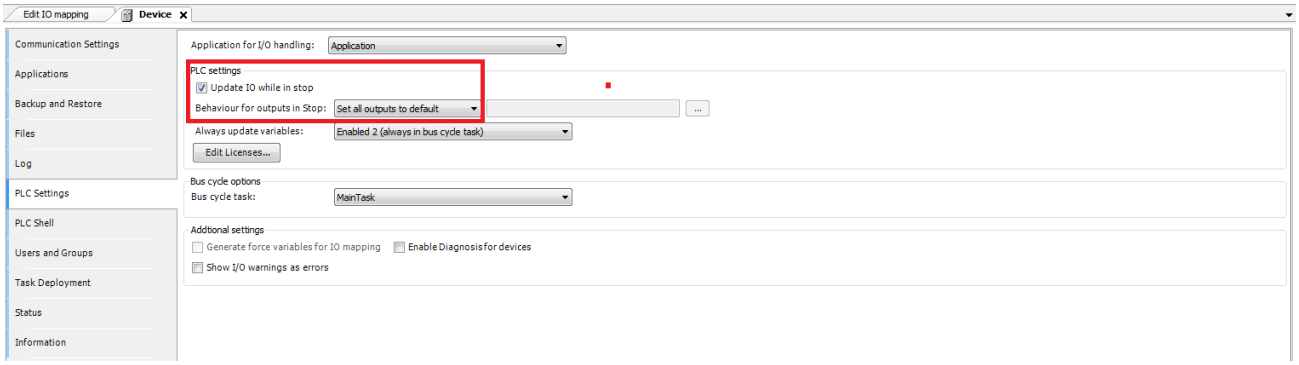


Рис. 14-б

В этом случае при переходе контроллера в стояние останова на модуле дискретных выходов не будет никакой индикации. При возвращении контроллера в рабочее состояние индикация на модуле примет состояние, соответствующее резервному режиму.

Важным результатом проведённых работ является заключение, что без доработки внутренней операционной системы не получается синхронизировать начало каждого SCAN-цикла в обоих контроллерах. Задача сделать так, чтобы в обоих контроллерах, в Основном и в Резервном, SCAN-циклы начинались одновременно с точностью до миллисекунд. Эта задача критична для систем типа Hot-Hot и Hot Standby. Для систем типа Warm Standby решение этой задачи необязательно.

Контроль состояния DIO сетей.

Принцип диагностики сети DIO основан на настройках каналов сканнера, считывающих данные с DIO-устройств, при которых в случае потери связи с DIO-устройством на стороне мастера данные устанавливаются в ноль, см. Рис.15.

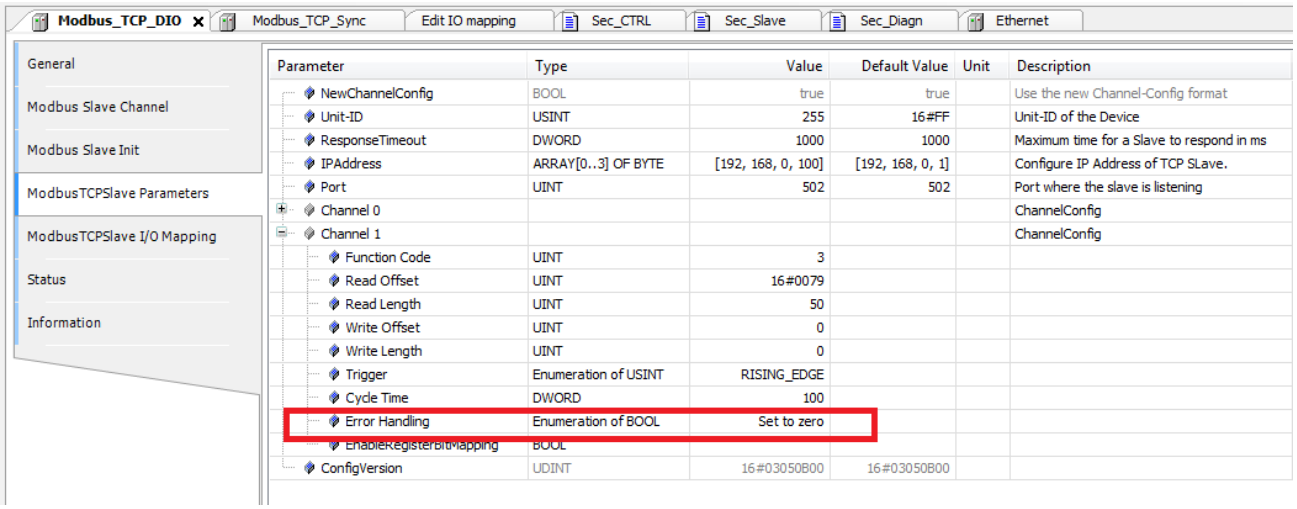


Рис. 15

Диагностика каждого DIO-устройства осуществляется с помощью разработанного нами функционального блока “DIO_Monitor”, см. Рис.16.

```
FUNCTION_BLOCK DIO_Monitor
(*-----------------------------------------------------------Мониторинг сети DIO 130423-140423-----------------------------------------------------------------------------------*)

//Триггер ошибок
IF NOT INIT THEN
TxTestWord:=16#FFFF;
TON_NetError(IN:=RxTestWord<>TxTestWord, PT:=T#1S);
SR_NetError(SET1:=TON_NetError.Q, RESET:=(ExtRST OR NOT TON_NetError.Q), Q1=>NetError);
END_IF;
//Счётчик ошибок
R_Trig_NetError(CLK:=TON_NetError.Q);
CTU_NetError(CU:=R_Trig_NetError.Q, RESET:=ExtRST, CV=>NetErrorNum);
//HealthBit
IF NOT TON_NetError.Q THEN HealthBit:=TRUE; ELSE HealthBit:=FALSE; END_IF;

(*-------------------------------------------------------------------------------Конец блока-----------------------------------------------------------------------------------*)
```

Рис.16

Контроллер посылает в режиме записи в каждое DIO-устройство определённое контрольное слово, для нашего примера это 16#FFFF.
Когда мастер считывает данные из DIO-устройства, он в том числе в этом же пакете получает и это контрольное слово. Далее происходит сравнение переданного и полученного значений.
В случае потери связи вместо контрольного слова мастер, в соответствии с настройками слейва, получит ноль. Если этот «ноль» присутствует более 1000 миллисекунд (на стенде использовалась эта величина), то взводится триггер неисправности канала “SR_NetError”, выходом которого является флаг ошибки “ NetError”. Инверсией флага ошибки является флаг исправности (HealthBit).
Для анализа и статистики ошибок каналов в программе используются счётчики фронтов флагов неисправностей каналов.
Триггер ошибки сети не может быть сброшен автоматически даже при условии восстановления сети. Для его сброса нужно задать команду через бит “ExternRST” от программатора или HMI/SCADA.
В функциональном блоке создан счётчик ошибок сети (“CTU_NetError”), значения которого можно выводить на HMI/SCADA.

Примечание. В данном примере не используется динамическое обновление контрольного слова с целью снижения нагрузки на сеть. При желании можно организовать счётчик посылок, мастер будет его посылать в DIO-устройство, считывать назад и сравнивать значения. При несовпадении значений взводить триггер ошибки сети.

Функциональный блок “DIO_Monitor”, вызывается в секции “Sec_DIO” в виде экземпляров с именами “Monitor_DIOxx” столько раз, сколько DIO-устройств используется в системе, см. Рис.17-а.

```
//Работа с DIO
BlinkDIO(ENABLE:=TRUE, TIMELOW:=T#500MS, TIMEHIGH:=T#500MS);
Tx_Array_DIO1[1].0:=BlinkDIO.OUT;
Tx_Array_DIO1[1].1:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].2:=BlinkDIO.OUT;
Tx_Array_DIO1[1].3:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].4:=BlinkDIO.OUT;
Tx_Array_DIO1[1].5:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].6:=BlinkDIO.OUT;
Tx_Array_DIO1[1].7:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].8:=BlinkDIO.OUT;
Tx_Array_DIO1[1].9:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].10:=BlinkDIO.OUT;
Tx_Array_DIO1[1].11:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].12:=BlinkDIO.OUT;
Tx_Array_DIO1[1].13:=NOT BlinkDIO.OUT;
Tx_Array_DIO1[1].14:=BlinkDIO.OUT;
Tx_Array_DIO1[1].15:=NOT BlinkDIO.OUT;

Tx_Array_DIO2[1].0:=BlinkDIO.OUT;
Tx_Array_DIO2[1].1:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].2:=BlinkDIO.OUT;
Tx_Array_DIO2[1].3:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].4:=BlinkDIO.OUT;
Tx_Array_DIO2[1].5:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].6:=BlinkDIO.OUT;
Tx_Array_DIO2[1].7:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].8:=BlinkDIO.OUT;
Tx_Array_DIO2[1].9:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].10:=BlinkDIO.OUT;
Tx_Array_DIO2[1].11:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].12:=BlinkDIO.OUT;
Tx_Array_DIO2[1].13:=NOT BlinkDIO.OUT;
Tx_Array_DIO2[1].14:=BlinkDIO.OUT;
Tx_Array_DIO2[1].15:=NOT BlinkDIO.OUT;
```

Рис.17-а.

В программной секции “Sec_DIO” имеется слово обобщённой ошибки связи со всеми DIO устройствами. Это слово “StatusWordDIO”. Каждый его бит соответствует по номеру ошибке связи с DIO-устройством с таким же номером. Если слово “StatusWordDIO” не равно нулю, то выставляется флаг готовности сети DIO (DIO_Ready), и на модуле SM3DQ16 программа зажигает светодиод LED9, см. Рис. 17-б.

```
//Обобщенная ошибка сети DIO
IF StatusWordDIO<>0 THEN DIO_Error:=TRUE; DIO_Ready:=FALSE; ELSE DIO_Error:=FALSE; DIO_Ready:=TRUE; END_IF;
```

Рис. 17-б.

В случае обнаружения ошибки хотя бы в одном DIO-устройстве флаг готовности сети DIO сбрасывается, устанавливается флаг обобщённой ошибки сети DIO (DIO_Error), и контроллер переводится в состояние “Offline”, на модуле SM3DQ16 гаснет светодиод LED9, и зажигается светодиод LED3.

Примечание. Этот принцип можно легко изменить по желанию Заказчика, например реализовать принцип, когда Основной контроллер будет оставаться основным, пока у него есть связь хотя бы с одним удалённым устройством.

Работа с HMI/SCADA

В документации на линейку SM250 указано, что количество подключений для Ethernet TCP равно 32.
На стенде проводилось тестирование следующих конфигураций:
-К контроллеру SM252 (сервер) подключалась один компьютер (клиент). Клиент читал из сервера 1000 Modbus регистров, что соответствует 8-ми Modbus-запросам, по 125 регистров каждый. Период опроса равен одной секунде.
Связь устойчивая, ошибок в сети нет. Пропусков данных нет.
-К контроллеру SM252 (сервер) подключалось два компьютера (клиента). Каждый клиент читал из сервера по 500 Modbus регистров, что соответствует 8-ми Modbus-запросам (8x125=1000 Modbus регистров). Период опроса равен одной секунде.
Связь устойчивая, ошибок в сети нет. Пропусков данных нет.
-К контроллеру SM252 (сервер) подключалось три компьютера (клиента). Каждый клиент читал из сервера по 256 Modbus регистров, что соответствует примерно 6-ти Modbus-запросам (6x125=750 Modbus-регистров). Период опроса равен одной секунде.
Связь устойчивая, ошибок в сети нет. Пропусков данных нет.

Дальнейшее увеличение количества клиентских компьютеров или увеличение количества запросов приводит к появлению пропусков данных. Данные идут, ошибки в сети не возникают, но стабильно имеют место пропуски запросов: через один, через два, и т.д. в зависимости от количества запросов и клиентов.
Примечание. В качестве Modbus TCP клиентов использовалась программа Modbus Poll=64 Bit V5.6.0, см. Рис.18-а:

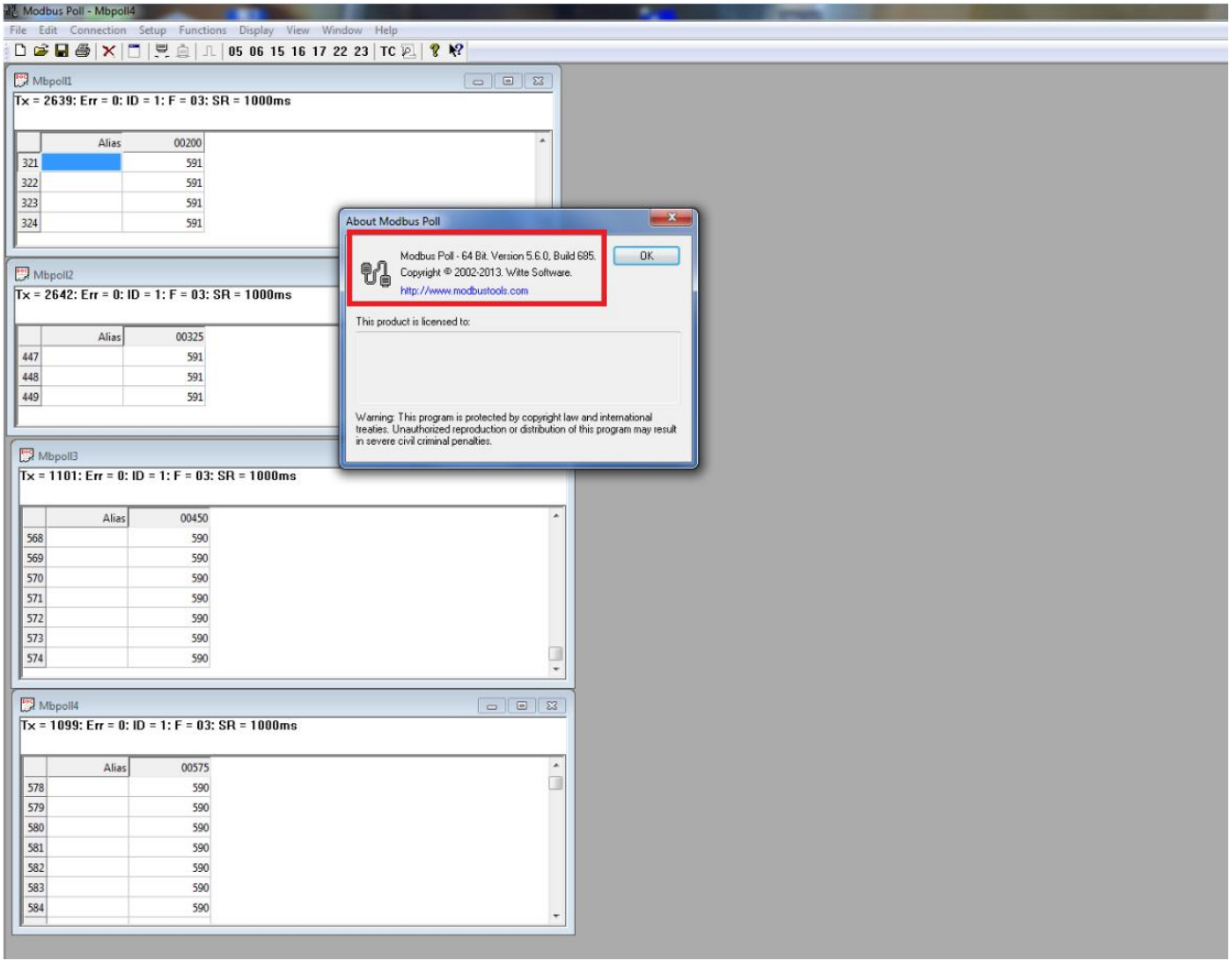


Рис.18-а

Проблему пропуска данных решили установкой в ноль параметра “Delay” функционального блока “MBUS_TCP_SLAVE”, Рис.18-б.
С одного сокета (один компьютер, один IP-адрес) программой Modbus Poll выдали запросы на 3000 регистров. Работает устойчиво, связь есть, замечаний нет.
То есть после коррекции входной переменной “Delay” функционального блока “MBUS_TCP_SLAVE” проблема не воспроизводится.

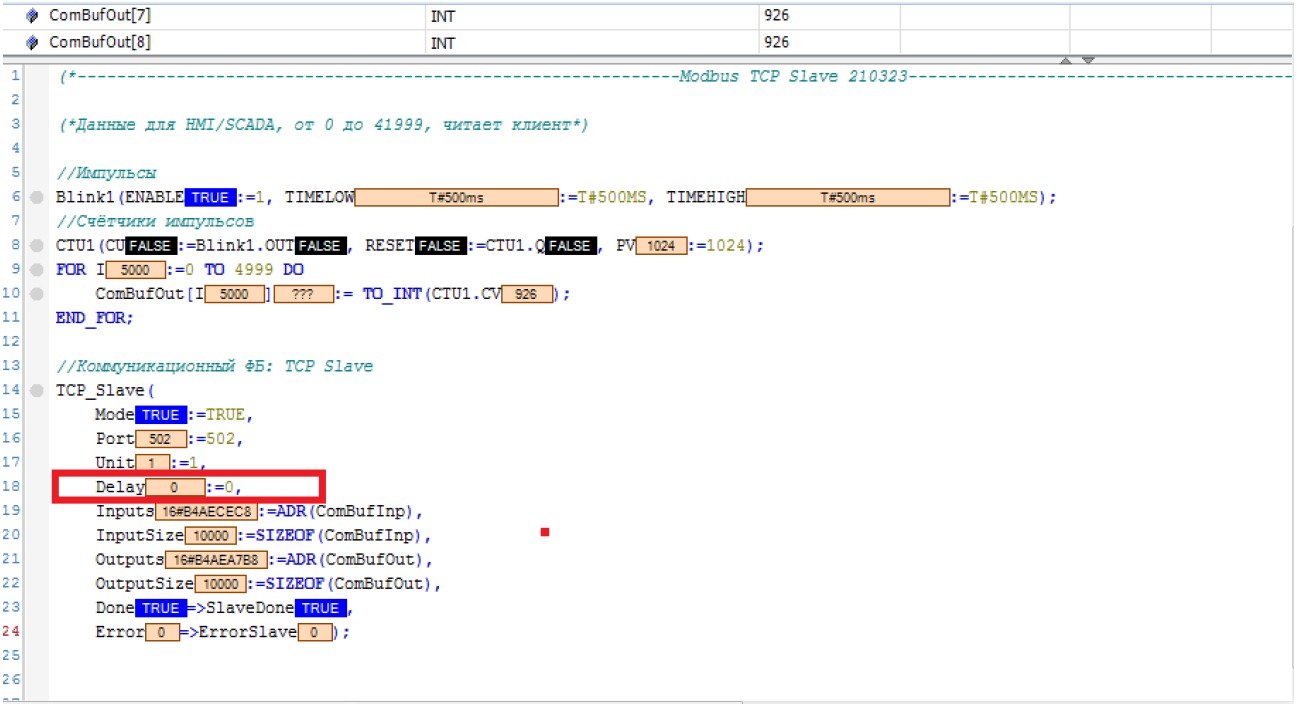


Рис.18-б.

Выполнено тестирование с несколькими сокетами. Для этого собрали систему: один ноутбук + 4 ПК (вне домена), итого 5 сокетов, на каждом запустили OFS на 5K Modbus регистров (INT).
Результат хороший, система отработала 1 час без единого сбоя и торможения. При этом на ПЛК работало еще программное резервирование и опрос 12 DIO устройств по 100-регистров. Загрузка ПЛК при этом была 4%.
Заключение: результат очень хороший.

Синхронизация системного времени.

В системах с резервированием возникает задача обеспечения одинакового системного времени в Основном и Резервном контроллерах.
В случае использования контроллеров платформы SM250 эту задачу невозможно решить использованием протокола NTP(SNTP), потому что этот протокол платформой не поддерживается.
Поэтому пришлось решать эту задачу используя библиотечные ресурсы среды разработки Codesys.
Для этого используются стандартные функциональные блоки “RTCLC.GetDateAndTime” и “RTCLC.SetDateAndTime”, см. Рис.19-а.

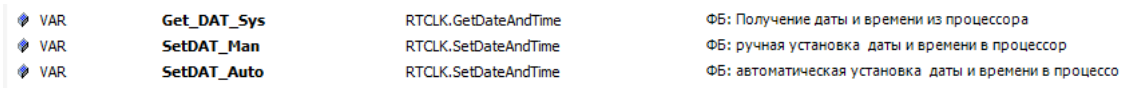


Рис.19-а

На базе этих стандартных блоков нами разработан функциональный блок “DAT”, см. Рис.19-б:

```
FUNCTION_BLOCK DAT
(*------------------------------------------ Блок: Установка и синхронизация даты и времени 140423-----------------------------------------------------------*)

(*Период записи и чтения даты и времени*)
BLINK1(ENABLE:=TRUE, TIMELOW:=Period, TIMEHIGH:=Period);
R_Trig_BLINK1(CLK:=BLINK1.OUT);

(*Ручная установка даты и времени*)
R_Trig_ManSet(CLK:=ManSet);
SetDAT_Man(xExecute:=R_Trig_ManSet.Q, dtDateAndTime:=ManDAT);

(*Ститывание даты и время процессора*)
Get_DAT_Sys(xExecute:=R_Trig_BLINK1.Q);
IF Get_DAT_Sys.xDone THEN TimePLC:=Get_DAT_Sys.dtDateAndTime; END_IF;

(*Запись даты и времени из основного контроллера в синхромассив*)
R_Trig_ThisPRM(CLK:=ThisPRM);
IF ThisPRM THEN
   DW_AS_WD1(Inp_DW:=DT_TO_DWORD(TimePLC));
   OutWord1:=DW_AS_WD1.Out_W0;
   OutWord2:=DW_AS_WD1.Out_W1;
END_IF;

(*Запись даты и времени в резервный контроллер из синхромассива*)
R_Trig_ThisSTB(CLK:=ThisSTB);
IF ThisSTB THEN
    WD_AS_DW1(Inp_W0:=TO_WORD(InpWord1), Inp_W1:=TO_WORD(InpWord2));
    SetDAT_Auto(xExecute:=R_Trig_BLINK1.Q OR R_Trig_ThisSTB.Q, dtDateAndTime:=DWORD_TO_DT(WD_AS_DW1.Out_DW));
END_IF;


(*-----------------------------------------------------------------Конец Блока-----------------------------------------------------------------------------*)
```

Рис. 19-б

Внутри блока “DAT” задаётся период синхронизации времени CPU (системного времени), организована возможность ручной установки системного времени, взятие системного времени из CPU Основного контроллера и запись в CPU Резервного контроллера системного времени, полученного из CPU Основного.
Информация о времени передаётся в составе синхромассива.

Функциональный блок “DAT” вызывается в программной секции “Sec_CTRL”, см. Рис. 19-в для Основного контроллера и 19-г для Резервного.

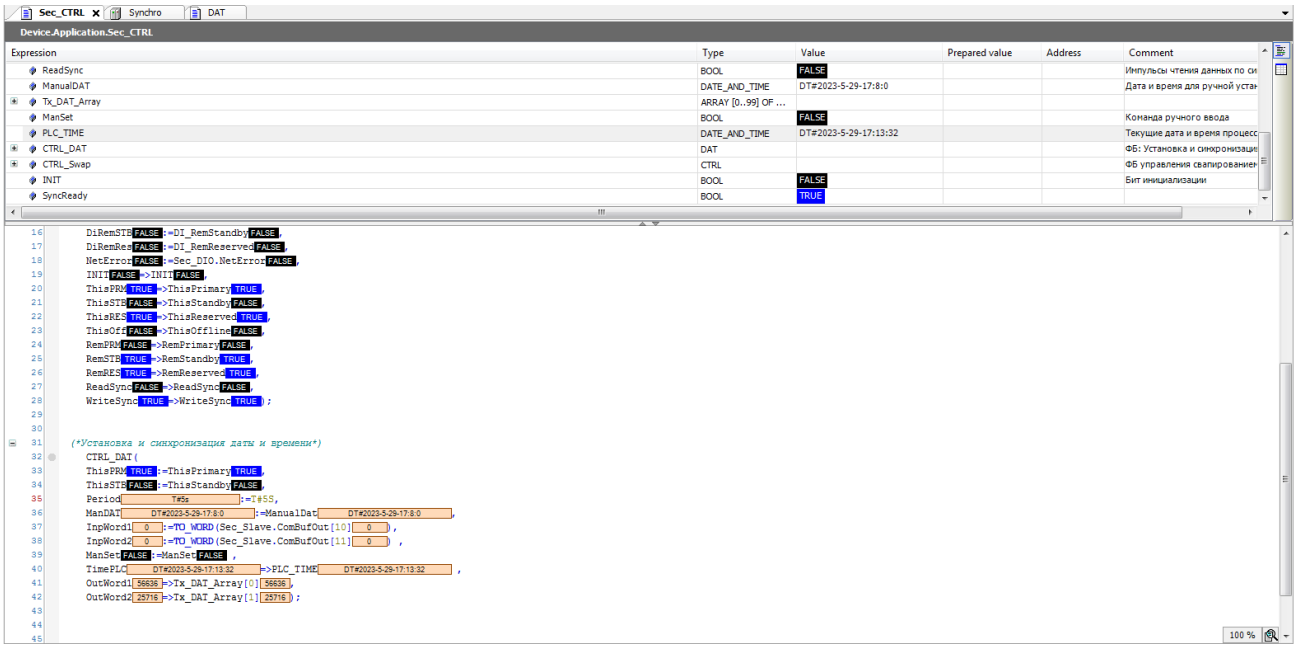


Рис. 19-в

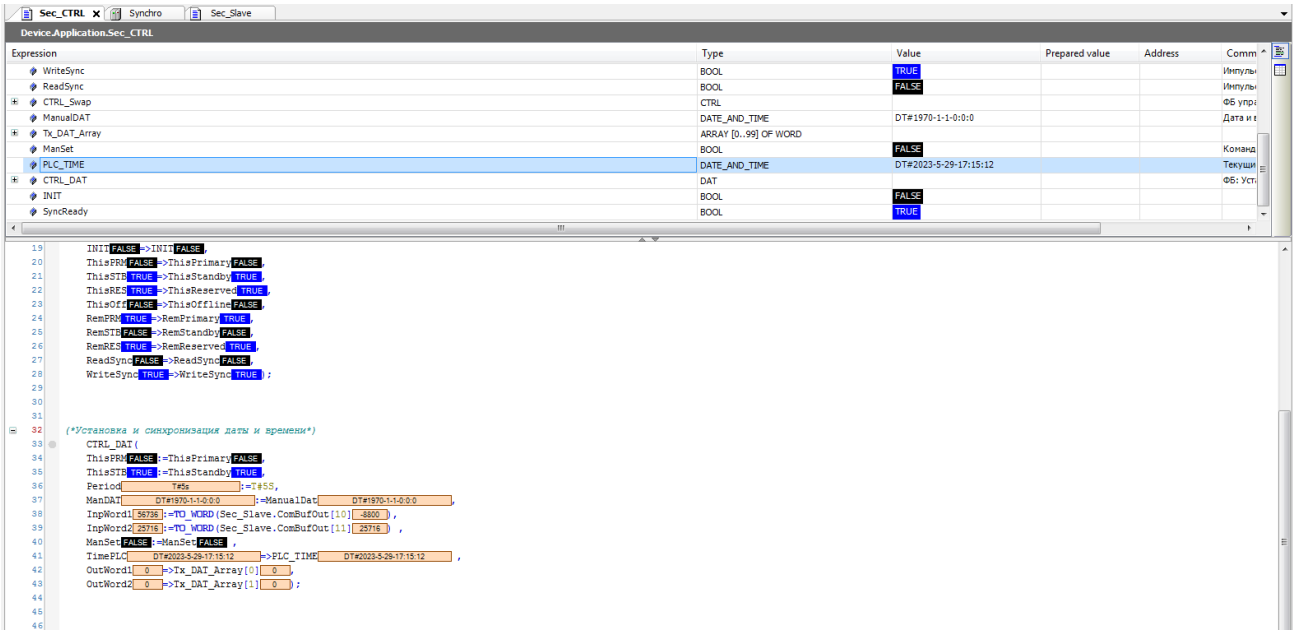


Рис. 19-г

Примечание. Данное решение предусматривает синхронизацию времени Основного и Резервного контроллеров с точностью до секунды.

Итоговое заключение

В результате данной работы было разработано и испытано решение, позволяющее создавать резервированные системы на основе контроллеров Systeme Electric SM252MESC (внутренняя операционная система V1.04), работающих в среде разработки приложений Codesys V3.5 SP11 Patch6. Конструктивно данная платформа не предназначена для построения систем с резервированием, так как имеет ряд аппаратных ограничений, поэтому построить полноценную систему с горячим резервированием (Hot-Hot или Hot- Standby) не получилось. Поэтому все функционалы резервирования были реализованы программно с помощью среды разработки приложений Codesys.
Для того, чтобы система на базе платформы SM250 была оценена как имеющая горячее резервирование, в дополнение к решенным в результате данной работы задачам необходимо:
-реализовать синхронизацию системных часов Основного и Резервного контроллера на базе протокола NTP (SNTP);
-реализовать переключение IP-адресов в контроллерах в соответствии с их статусом, полученным при вводе резерва;
-обеспечить синхронизацию начала и конца циклов в Основном и Резервном контроллерах в процессе работы;
-обеспечить аппаратную диагностику сетей управления и синхронизации на уровне каждого канала;
-обеспечить возможность выравнивания программ в контроллерах без остановки процесса управления при внесении изменений в один из них.
Для решения этих задач необходимы следующие решения со стороны Вендора.
-Расширение аппаратной части платформы за счёт добавления коммуникационных модулей, имеющих статус сопроцессора (коммуникационного процессора). Минимум нужен ещё один такой модуль для разделения каналов синхронизации и управления. Для систем с большим объёмом обмена данными между контроллером и HMI/SCADA желательно использование ещё одного коммуникационного модуля для связи с системой верхнего уровня.
-Доработка внутренней операционной системы процессорных модулей согласно выданного нами задания.

Рассмотренное в данном документе решение может быть применено только Систем с Теплым Резервированием (Warm Standby).

Примечание. Проект (программа на Codesys) пока выполнена на основе созданных нами пользовательских функциональных блоков, см. Рис.20-а:

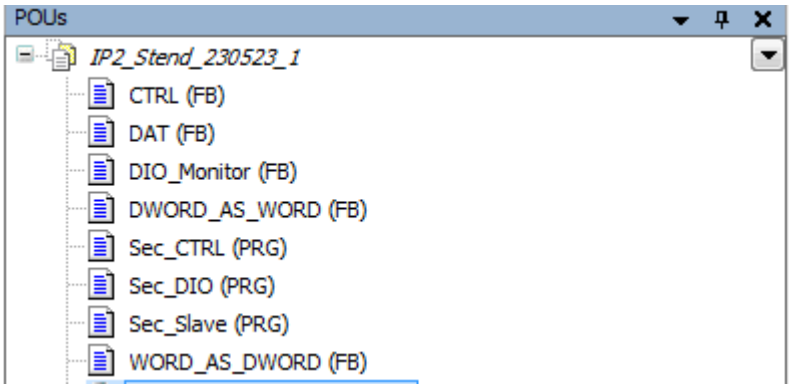


Рис.20-а

При необходимости, если станет вопрос о тиражировании решения, то эти блоки могут быть трансформированы в формат библиотек Codesys, см. Рис.20-б:

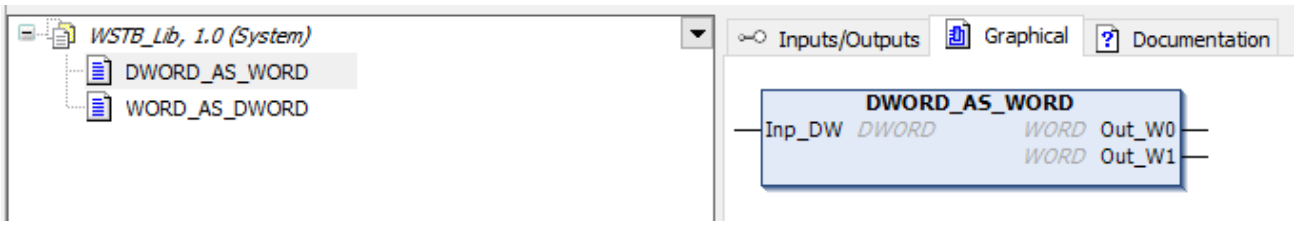


Рис.20-б

Systeme Electric. Апрель-Май-Июнь 2023 года.