**ШИШОВ О. В., ГЕРАСЬКИН Е. В.**

# РАСПРЕДЕЛЕНИЕ АДРЕСОВ ПЕРЕМЕННЫХ В ПОЛЕ ДАННЫХ ПРОТОКОЛА MODBUS

**Аннотация**. Статья посвящена рассмотрению примера программирования логического контроллера ОВЕН PLC100 в среде CoDeSys и операторской панели ОВЕН ИП320 в среде «Конфигуратор ИП320», поясняющего принципы организации сетевого обмена между этими устройствами по протоколу ModBus.

**Ключевые слова**: контроллер, операторная панель, интерфейс, ModBus.

**SHISHOV O. V., GERASKIN E.V.**

# ALLOCATION OF VARIABLES' ADDRESSES IN MODBUS PROTOCOAL DATA FIELD

**Abstract**. The article discusses an example of programming logic controller OWEN PLC100 environment CoDeSys and operator panel OWEN ИП320 using the "Configurator ИП320" explains principles of network communication between the devices via ModBus.

**Keywords**: controller, operator panel interface, ModBus.

Современные системы управления промышленным оборудованием все чаще создаются исходя из принципа «распределенного интеллекта», когда они строятся из относительно самостоятельно работающих узлов (контролеров, операторных панелей, модулей ввода-вывода), обменивающихся между собой необходимой информацией по сети. При этом одной из задач проектировщика системы управления становится организация сетевого обмена. Многие производители подобного оборудования для расширения сфер применения своей продукции оснащают ее стандартными сетевыми интерфейсами. Так российская компания ОВЕН для организации сетевого взаимодействия многих своих устройств использует протокол ModBus.

Статья посвящена рассмотрению примера программирования логического контроллера ОВЕН PLC100 в среде CoDeSys и операторской панели ОВЕН ИП320 с помощью программы «Конфигуратор ИП320», поясняющего принципы организации сетевого обмена между этими устройствами по протоколу ModBus.

Весь объем передаваемой информации по протоколу ModBus (поле данных кадра) удобно рассматривать как некое поле памяти, разделенное на некоторое количество равных частей – регистров, каждый из которых имеет свой адрес. Каждый регистр имеет размер в 2 байта и в свою очередь состоит из двух частей по 8 бит. Мы можем использовать четыре типа переменных, каждая из которых занимает определенный объем памяти в поле памяти. Это переменные типа Byte – занимает объем в восемь бит, переменные типа WORD – 16 бит

(один регистр), типа DWORD –32 бита (два регистра) и типа REAL – используемые при отображении чисел с плавающей запятой, так же занимает объем в два регистра. При организации поля памяти, в зависимости от того, переменные какого типа мы используем в программе, выделяются участки памяти размером в один байт (адресация осуществляется побитно), один регистр (2 байта), или два регистра (4 байта). Введение в поле памяти переменных различных размеров требует определенного структурирования памяти – выравнивания переменных. Суть упорядоченного размещения переменных в области памяти или «выравнивания», принятого в среде CoDeSys, заключается в организации физической памяти таким образом, что однобайтные, двухбайтные и четырех байтные переменные должны располагаться только по определенным адресам. Адрес четырехбайтной переменной должен быть кратен четырем, двухбайтной – кратен двум, а однобайтной – кратен одному.

Для того чтобы разобраться в адресации поля памяти протокола рассмотрим небольшой пример. Предположим, перед нами стоит задача отразить на операторской панели ИП320 состояние трех дискретных датчиков, которые подключены к PLC100, вывести на операторскую панель значение двухбайтной переменной, задаваемой с контроллера (назовем ее disp) и записать в контроллер двухбайтную переменную (disp2), которую мы будем вводить с помощью клавиатуры операторской панели.

Программирование контроллера и операторной панели осуществляется отдельно. Перед началом этих работ в целом определимся, что в сетевом обмене контроллер будет играть роль подчиненного (slave) устройства, а операторная панель будет являться ведущим (master) устройством.

Прежде чем создать программу контроллера необходимо провести его конфигурацию, в рамках которой нужно связать его дискретные входы и выходы с переменными в программе, ввести сетевые элементы и настроить их, а также определить порядок расположения переменных, участвующих в обмене, в поле памяти протокола.

Пусть дискретные сигналы с датчиков поступают на три первых дискретных входа контроллера. Состояние этих входов будут определять в программе битовые переменные in1, in2 и in3. Предложим ввести в программу три переменные out1, out2 и out3, которые будут принимать значения соответственно переменных in1, in2 и in3. В конфигурации контроллера зададим, что значения переменных out1, out2 и out3 будут определять состояния трех первых дискретных выходов контроллера. Это позволит сделать наглядным работу программы контроллера, т. к. состояние его выходов отображается работой светодиодов на передней панели. После данных действий конфигурация ПЛК примет вид как на рисунке 1.

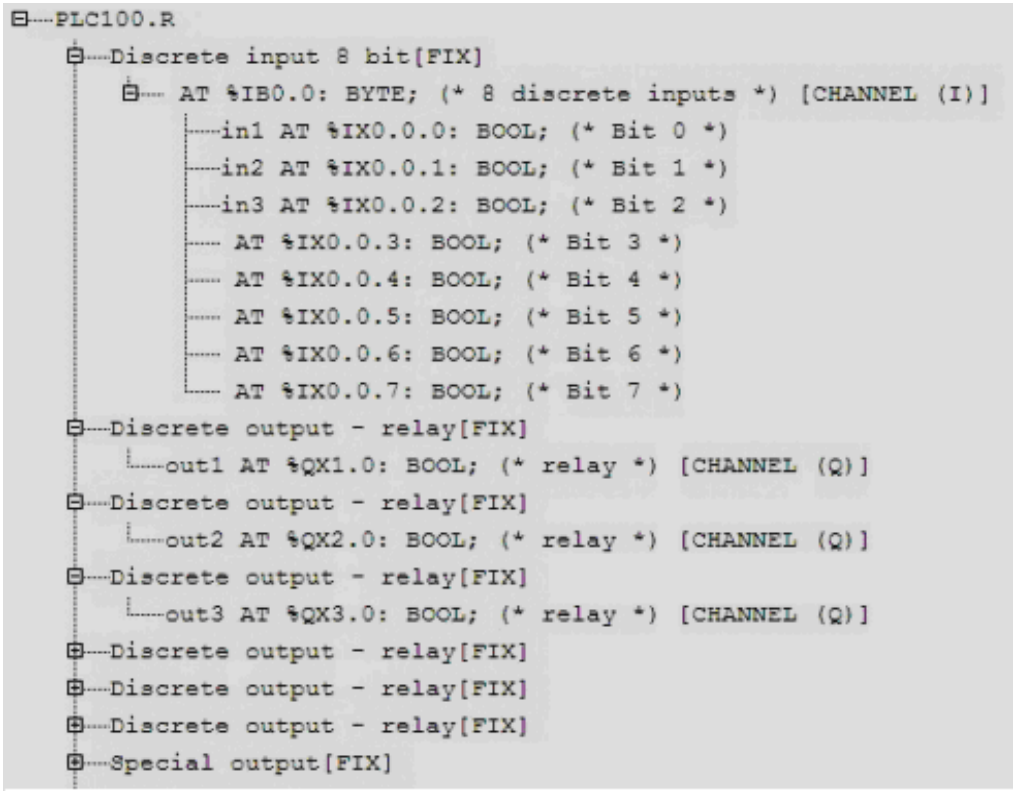


Рис. 1. Конфигурирование входов и выходов ПЛК в пакете CoDeSys для рассматриваемого примера

Введем в конфигурацию сетевые элементы, которые будут участвовать в обмене данными. Начнем с того, что в конфигурацию PLC добавим сетевой модуль ModBus (slave) и настроим параметры этого модуля (установим скорость обмена, наличие бита паритета и стоп бита, установим режим RTU и т. д.). Определим, что на физическом уровне будет использоваться интерфейс RS-485.

Далее введем в конфигурацию контролера (а точнее в конфигурацию его сетевого модуля) сетевые переменные, помня о том, что порядок их ввода скажется на порядке (адресах) их расположения в поле памяти интерфейса. В дальнейшем, при конфигурировании операторской панели, нам необходимо будет указывать эти адреса. Передаваться будут переменные key1, key2 и key3, которые в программе будут принимать значения соответственно переменных in1, in2 и in3.

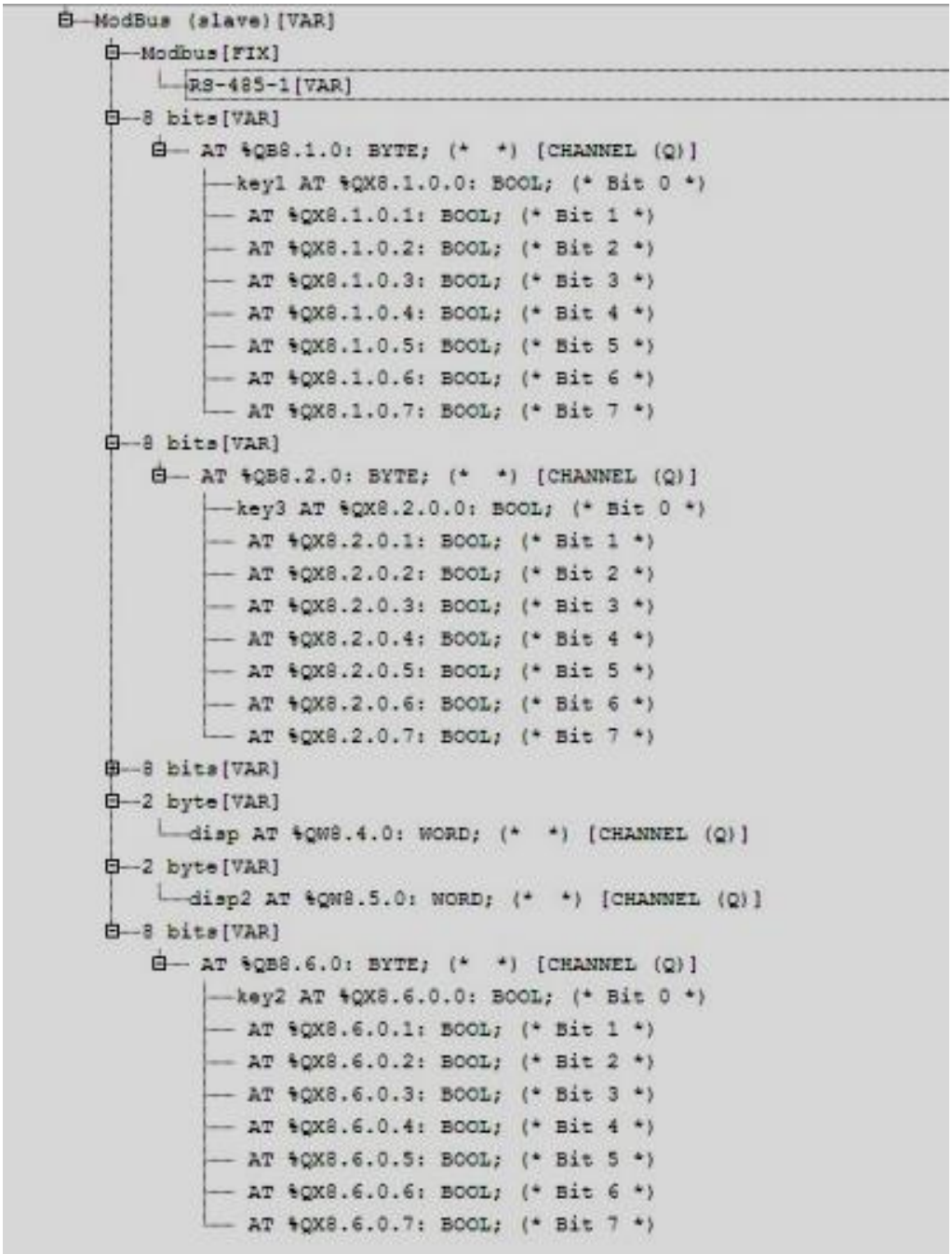


Рис. 2. Конфигурирование сетевых элементов в пакете CoDeSys для рассматриваемого примера

Для передачи битовой переменной key1 в сетевую конфигурацию введен элемент «8-бит». Примем, что состояние переменной key1 будет отражать нулевой бит этого байта. Для передачи битовой переменной key2 в сетевую конфигурацию введен еще один элемент «8-бит». Примем, что состояние переменной key2 будет отражать нулевой бит этого байта.

Безусловно, для передачи двух битовых переменных можно обойтись введением в конфигурацию одного байта. Но демонстрируемый пример составляется так, чтобы наиболее

полно отразить особенности формирования адресов переменных в поле памяти протокола. С этой же целью введем в сетевую конфигурацию третий элемент «8-бит». Будем считать, что он зарезервирован для передачи каких-то переменных еще не введенных в программу на этом этапе ее создания.

Далее в конфигурацию введем два элемента «2-byte». Первый из них будет использоваться для передачи переменной disp, второй – для передачи переменной disp2.

Наконец введем в конфигурацию еще один элемент «8-бит», нулевой бит этого байта будет отражать состояние переменной key3.

Как будет выглядеть конфигурирование сетевых элементов в пакете CoDeSys, показано на рисунке 2. При этом структура поля памяти интерфейса с указанием адресов битов, байтов, регистров для используемых переменных показано в Таблице 1.

Таблица 1.

Размещение сетевых переменных в поле памяти протокола ModBus

| **Номер бита** | **Номер байта** | **Расположение переменных в памяти** | **Адрес регистра** |
|---|---|---|---|
| 0…7 | 0 | key 1 | 0 |
| 8…15 | 1 | key 2 | |
| 16…23 | 2 | Зарезервированная ячейка | 1 |
| 24…31 | 3 | Выравнивание переменных | |
| 32…39 | 4 | disp | 2 |
| 40…47 | 5 | | |
| 48…55 | 6 | disp 2 | 3 |
| 56…63 | 7 | | |
| 64…71 | 8 | key 3 | 4 |
| 72…79 | 9 | | |

Первый элемент «8-бит» занимает первый байт регистра с адресом «ноль», второй элемент «8-бит» – второй байт этого регистра, третий элемент «8-бит» – занял первый байт регистра с адресом 1, второй байт регистра с адресом 1 не используется переменной disp, так как структурирование памяти ведется по регистрам, и переменой disp достается полностью регистр с адресом 2, а переменной disp2 – регистр с адресом 3, четвертый элемент «8-бит» занял первый байт регистра с адресом 4.

При задании адреса битовой переменной в поле памяти протокола указывается номер бита без учета деления памяти на байты и регистры, поэтому в дальнейшем при

конфигурировании операторной панели адрес битовой переменной key1 будет указываться равным 0, переменной key2 в поле памяти будет соответствовать бит с адресом 8, переменной key3 – бит с адресом 48.

Адрес двухбайтной переменной в поле памяти протокола задается номером регистра.

Рисунок 3 показывает программу работы контроллера, созданную для этого примера. Программа создана в пакете CoDeSys на языке CFC. Программа создавалась лишь для демонстрации рассматриваемого примера, поэтому является максимально простой. Сигнал, соответствующий уровню логической единице или нуля, подаваемый на вход контроллера in1, будет передаваться на выход контроллера out1. Сигнал, подаваемый на вход in2, будет передаваться на выход out2. Сигнал, подаваемый на вход in3, будет передаваться на выход out3. Одновременно по значениям входных сигналов будут формироваться значения сетевых переменных key1, key2 и key3. Значению переменной disp присваивается значение 100, переменной p при работе программы будет присваиваться значение переменной disp2.

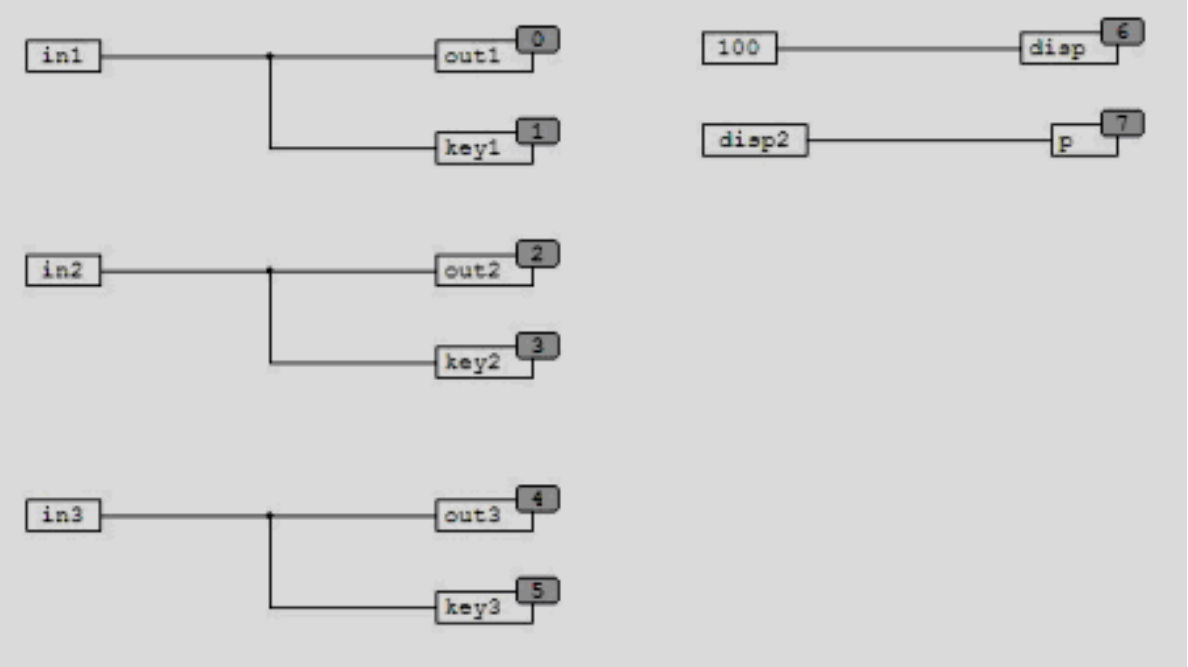


Рис. 3. Пример программы PLC100

Перейдем к конфигурированию операторской панели ИП320 с помощью программы «Конфигуратор ИП320». Прежде всего, в конфигураторе нам нужно осуществить общие сетевые настройки – указать, что в сетевом обмене панель будет являться ведущим (master) устройством, выставить скорость обмена данными (она устанавливается та, которую мы указали при конфигурировании сети контроллера). Далее нам нужно указать способ отображения данных на экране и указать адреса, по которым операторская панель будет

устанавливать или брать значения соответствующих данных в поле памяти протокола ModBus. Эти адреса нам известны – мы их определили (фактически задали) ранее, задавая конфигурацию сетевых элементов контроллера.

Используя знание адресов в соответствии с приведенными выше рассуждениями об адресах в поле памяти используемых нами переменных, проводим конфигурирование панели так, как это последовательно показано ниже.

Для отображения переменных out1, out2 и out3 выбран элемент «Лампа» – в зависимости от значения переменной на экране операторной панели она будет менять свой цвет. Рисунок 4 показывает, где на экране будет размещаться «Лампа» соответствующая переменной out1 и определение ее адреса в поле переменной. Этот адрес равен 0, т. е. это нулевой по счету бит в поле памяти протокола.

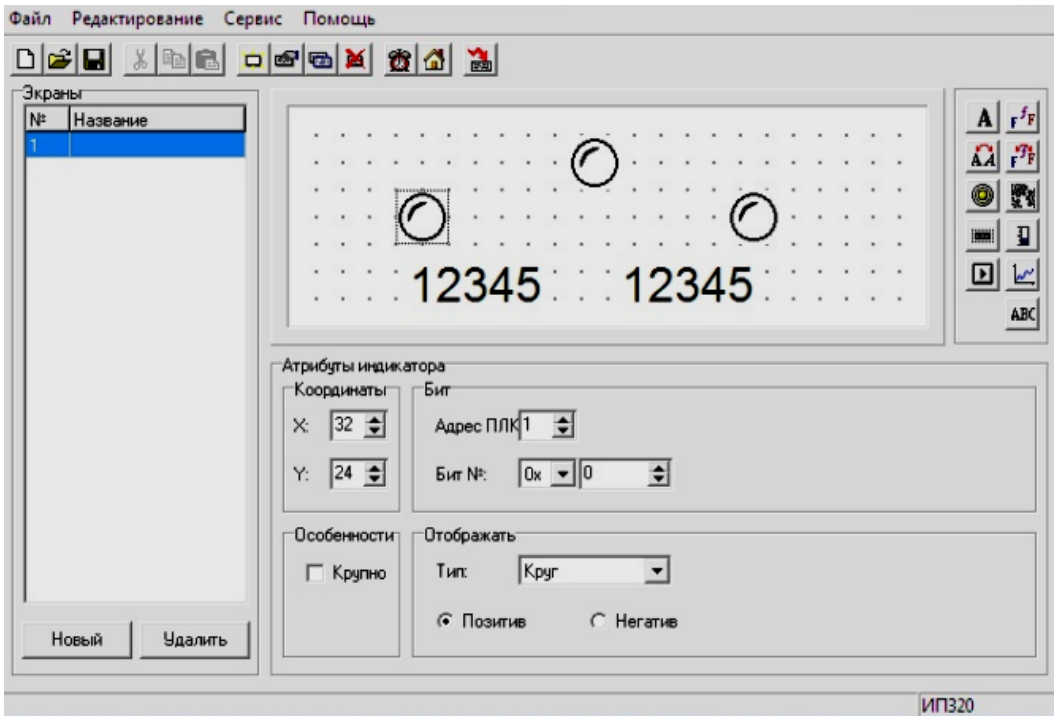


Рис. 4. Конфигурирование элемента «Лампа» для отображения состояния первого дискретного датчика

Рисунок 5 показывает, где на экране будет размещаться «Лампа» соответствующая переменной out2 и определение ее адреса в поле переменной. Этот адрес равен 8, т. е. это восьмой по счету бит в поле памяти протокола. Рисунок 6 показывает, где на экране будет размещаться «Лампа» соответствующая переменной out3 и определение ее адреса в поле переменной. Этот адрес равен 64, т. е. это шестьдесят четвертый по счету бит в поле памяти протокола.

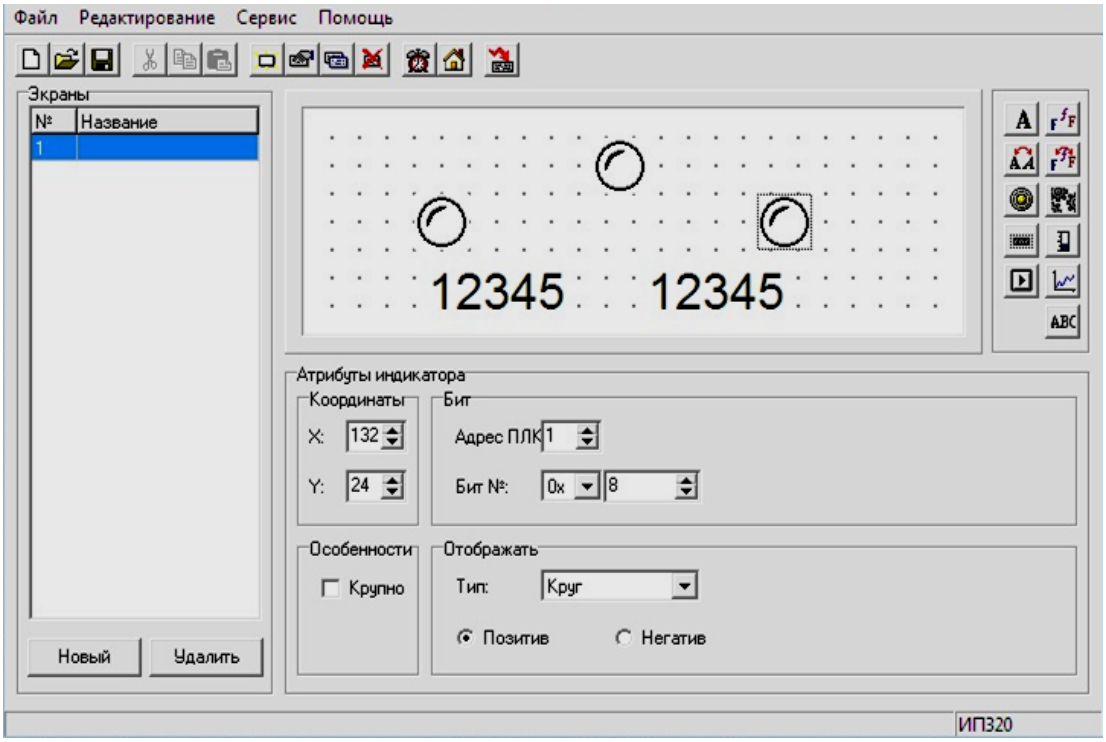


Рис. 5. Конфигурирование элемента «Лампа» для отображения состояния второго дискретного датчика

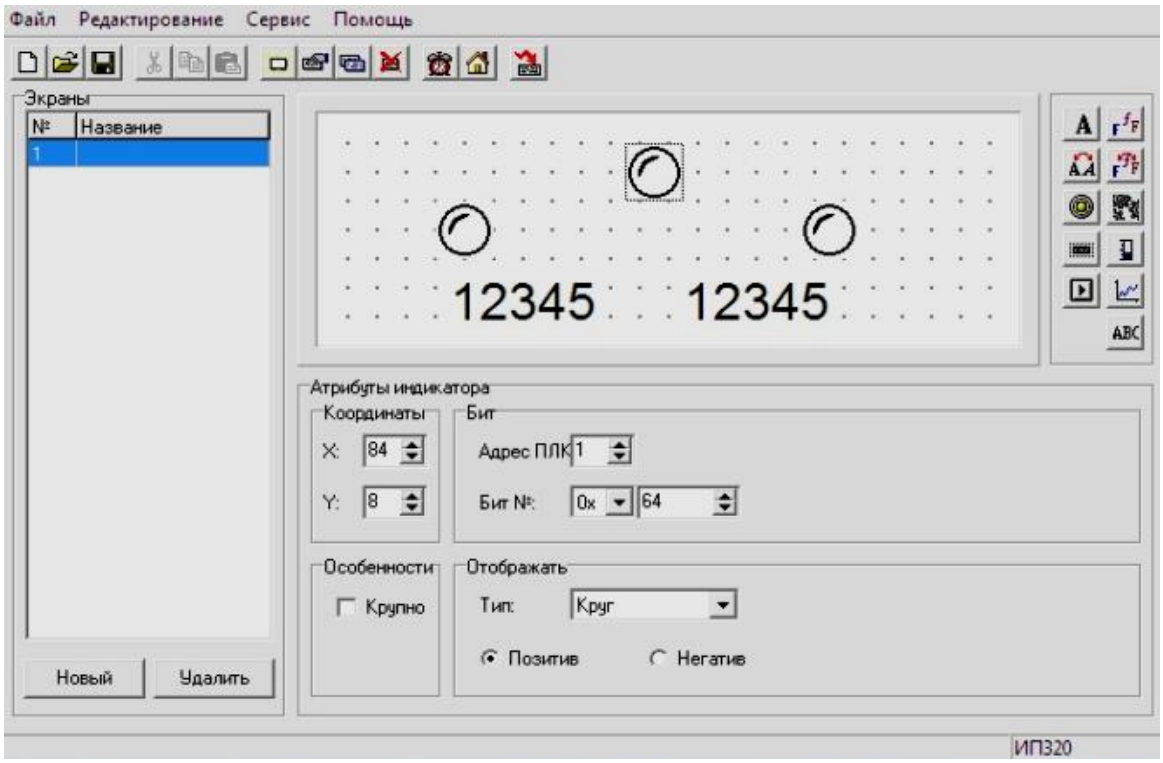


Рис. 6. Конфигурирование элемента «Лампа» для отображения состояния третьего дискретного датчика

Для отображения переменных disp и disp2 выбран элемент «Дисплей» – на экране операторной панели эти элементы будут отражать значения переменных в цифровой форме. Рисунок 7 показывает, где на экране будет размещаться «Дисплей», соответствующий переменной disp и определение ее адреса в поле переменной. Этот адрес равен 2, т. е. это второй регистр в поле памяти протокола.

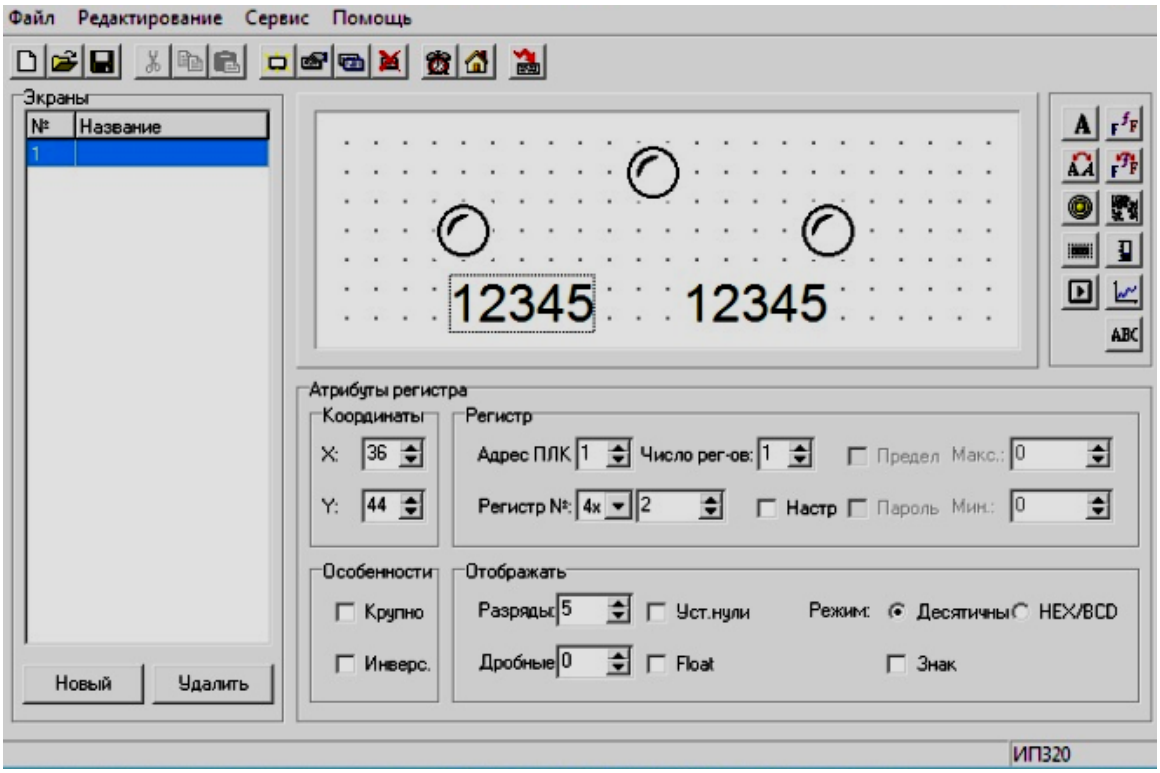


Рис. 7. Конфигурирование элемента «Дисплей» для вывода на ИП320 значения двухбайтной переменной

Рисунок 8 показывает, где на экране будет размещаться «Дисплей», соответствующий переменной disp2 и определение ее адреса в поле переменной. Этот адрес равен 3, т. е. это третий регистр в поле памяти протокола.

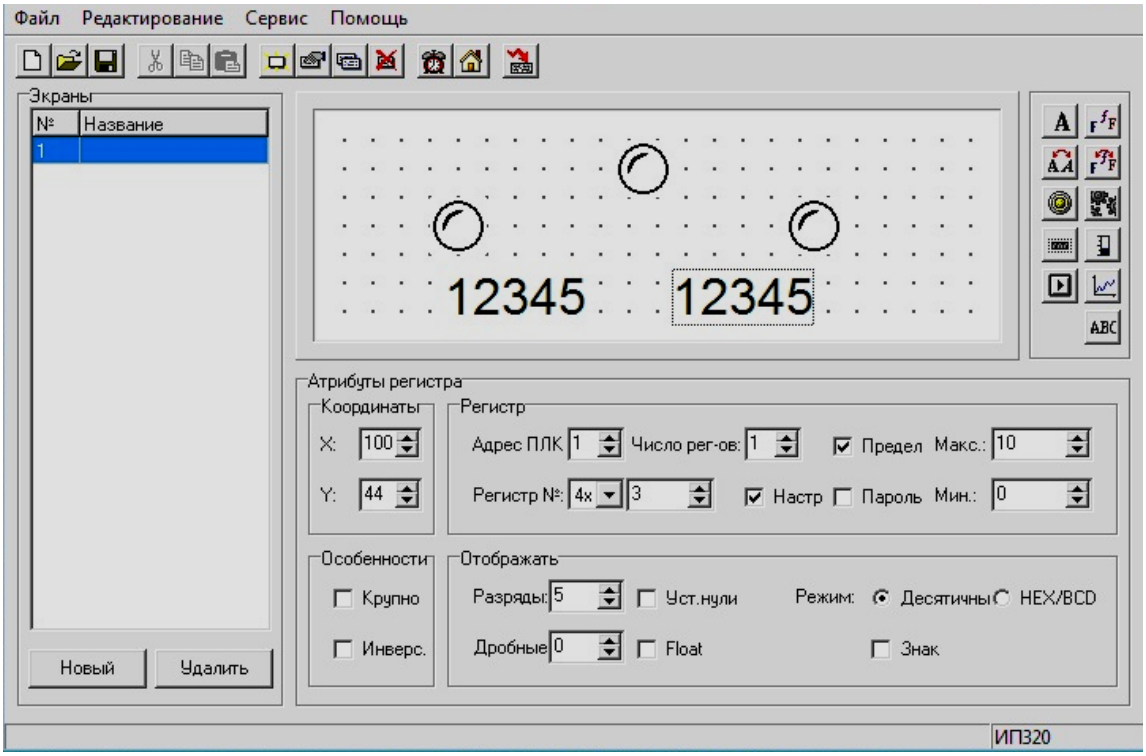


Рис. 8. Конфигурирование элемента «Дисплей» для ввода значения двухбайтной переменной с ИП 320

Осуществив загрузку программы контроллера в контроллер и проекта визуализации в операторную панель, соединив их между собой сетевым кабелем необходимо убедиться в их совместной работе.

Данный пример наглядно показывает основные принципы работы с адресами переменных в поле данных кадра при организации сетевого взаимодействия по протоколу ModBus. Пример разрабатывался для учебных стендов лаборатории «Современные системы промышленной автоматизации» кафедры электроники и наноэлектроники Мордовского госуниверситета им. Н. П. Огарева.

ЛИТЕРАТУРА

1. Шишов О. В. Современные технологии промышленной автоматизации : учеб. – Саранск Изд-во Мордов. ун-та,, 2009. – 276 с.

2. Шишов О. В. Технические средства автоматизации и управления : учеб. пособие. – М.: ИНФРА-М, 2011. – 397 с.

3. Программируемые логические контроллеры: ОВЕН ПЛК. – [Электронный ресурс] – Режим доступа: www.owen.ru