Министерство образования и науки Российской Федерации

Амурский государственный университет

А.Н. Рыбалев

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ И АППАРАТУРА УПРАВЛЕНИЯ: ЛАБОРАТОРНЫЙ ПРАКТИКУМ. ЧАСТЬ 4. СИСТЕМЫ УПРАВЛЕНИЯ ЛИФТОМ

Благовещенск

Издательство АмГУ

2014

ББК 32.965 я 73
Р93

Рекомендовано
учебно-методическим советом университета

Рецензенты:

Ю.В. Мясоедов, декан энергетического факультета АмГУ, профессор, канд. техн. наук;

Е.П. Ялама, главный инженер управления пуско-наладочных работ ОАО «Гидроэлектромонтаж»

Р93 Рыбалев А.Н. Программируемые логические контроллеры и аппаратура управления: лабораторный практикум. Часть 4. Системы управления лифтом. Учебное пособие. – Благовещенск: Амурский гос. ун-т, 2015. – 96 с.

Пособие предназначено для студентов направления 220700.62 «Автоматизация технологических процессов и производств», изучающих дисциплины «Средства автоматизации и управления», «Программное обеспечение систем управления», «Теория дискретных систем управления», «Автоматизация технологических процессов» и выполняющих лабораторные работы по данным дисциплинам. Может быть также использовано при выполнении курсовых и дипломных проектов.

Автор выражает признательность В.А. Чупрову, А.С. Сергееву, С.В. Ванзину, А.Е. Комарову, А.А. Чукину, Р.В. Брусник, Р.Д. Редозубову, В.П. Харькову.

В авторской редакции.

СОДЕРЖАНИЕ

Благодарности

Автор выражает благодарность

выпускникам Амурского государственного университета специальности 220301 «Автоматизация технологических процессов и производств»:

Чупрову Виктору Анатольевичу, *Сергееву Артему Сергеевичу* – проектировщикам и непосредственным создателям физической модели лифтового механизма (2010 год);

Ванзину Сергею Владимировичу, Комарову Андрею Евгеньевичу – первым разработчикам системы управления и программного обеспечения стенда (2011 год);

Чухину Артему Александровичу, проделавшему большой объем работ по модернизации стенда в 2012 году;

Брусник Роману Владимировичу, разработавшему и протестировавшему комплекс лабораторных работ на стенде в 2013 году;

Редозубову Роману Дмитриевичу, бывшему старшему преподавателю кафедры автоматизации производственных процессов и электротехники Амурского государственного университета, автору идеи стенда, многолетнему руководителю и консультанту курсовых и дипломных проектов по его разработке и модернизации;

Харькову Владимиру Пантелеевичу, высококвалифицированному рабочему кафедры автоматизации производственных процессов и электротехники Амурского государственного университета, главному исполнителю всех монтажных работ.

Физическая модель лифта

Модель лифта показана на рис. 1.

На рисунке обозначены:
1 – шахта лифта;
2 – трос;
3 – кабина лифта;
4 – двери кабины;
5 – привод дверей;
6 – направляющие дверей;
7,8 – направляющие дверей;
9 – блок питания;
10 – шкив;
11 – редуктор;
12 –двигатель главного привода;
13 – аварийный выключатель;
14 – основание;
15 – пускатель аварийный;
16 – фотодиод (этажные и межэтажные датчики);
17 – светодиод;
18 – шина проводов;

Рис. 1. Физическая модель лифта.

Размеры установки были определены исходя из возможностей лаборатории. Высота шахты равняется 2,9 м (при высоте потолка в 3,2 м). Шахта представляет собой две параллельные направляющие, выполненные из стального уголка. Шахта прикрепляется как к основанию, на котором размещены другие элементы установки, так и к потолку лаборатории.

Количество этажей принято равным четырем. Межэтажное расстояние составляет 0,65 м, высота кабины – 0,45 м. Кабина снабжена боковыми конструкциями, включающими подшипники качения, посредством которых она удерживается и передвигается по направляющим шахты.

Двигатель главного привода 4АА50В4У3 имеет синхронную скорость вращения 1500 об/мин, номинальную мощность 0,29 кВт. Передаточное число червячно-цилиндрического редуктора равно 167.На выходной вал редуктора насажен шкив, наматывающий трос, соединенный с кабиной лифта через блоки, установленные на верхних перекрытиях шахты.

На максимальной скорости кабина проходит от первого этажа до четвертого за 15 с. Двигатель питается от преобразователя частоты ASC 300, с помощью которого осуществляется регулирование скорости подъема/опускания лифта.

Двери кабины имеют собственный привод на основе двигателя постоянного тока. Для определения состояния дверей предусмотрены три датчика (двери открыты, закрыты или находятся под внешним воздействием).

Положение кабины контролируется оптическими датчиками. Передатчик всех датчиков установлен на крыше кабины, а приемники расположены на шахте лифта. Этажные приемники отвечают за точную остановку кабины на этаже. Промежуточные – за снижение частоты вращения привода при подходе к требуемому этажу. Сигналы датчиков усиливаются транзисторными усилителями. Отдельная оптическая пара установлена на валу двигателя и формирует два импульса на один его оборот. Этот сигнал может использоваться вместо сигналов межэтажных датчиков.

В конструкции установки также предусмотрено автоматическое отключение привода в тех случаях, когда по каким-либо причинам кабина лифта опустится ниже первого этажа или поднимется выше четвертого. Аварийное отключение осуществляют конечные выключатели, расположенные в верхней и нижней частях одной из направляющих шахты.

На пульте управления расположены кнопки *приказов* и *вызовов*, а также светодиоды для фиксации фактов их принятия системой управления. Для второго и третьего этажа предусмотрены по две кнопки вызова: вниз и вверх.

Система управления лифтом

Структура системы управления показана на рис. 2.

Рис. 2. Структура системы управления.

Основной привод стенда управляется частотным преобразователем ACS 300, который обеспечивает плавный разгон и плавное торможение кабины. Общее управление установкой осуществляет программируемый логический контроллер Siemens S7-226.

Контроллер получает сигналы следующих датчиков:

1) кнопки пульта;
2) концевые выключатели дверей и датчик перегрузки;
3) этажные и межэтажные датчики положения кабины;
4) датчик на валу двигателя.

Выполняя программу, Siemens S7-226 формирует сигналы управления:

сигналы управления 1 (преобразователем частоты):

1) пуск;

2) реверс;

3) переход на пониженную скорость;

сигналы управления 2 (непосредственно моделью лифта):

1) включение привода дверей кабины на их открытие;

2) включение привода дверей кабины на их закрытие;

3) сигналы управления светодиодной индикацией пульта.

Принципиальная схема соединений входных цепей контроллера показана на рис. 4.

На рис. 4 обозначены:

SB1–SB4 – кнопки приказов на пульте управления;

SB5–SB10 – кнопки вызовов кабины на пульте управления;

SQ1–SQ3 –концевые выключатели, использующиеся в качестве датчиков состояния дверей (дверь закрыта, находится под внешней нагрузкой, закрыта);

VD1 –VD10 – фотодиоды датчиков положения кабины в шахте: VD1, VD4, VD7, VD10 – этажные датчики, остальные –межэтажные;

VD13 – светодиод датчиков положения кабины в шахте;

VD11, VD12 – оптическая пара датчика оборотов двигателя главного привода.

Для усиления сигналов всех оптических датчиков используются одинаковые блоки усиления на биполярных транзисторах, их схема показана на рис. 3.

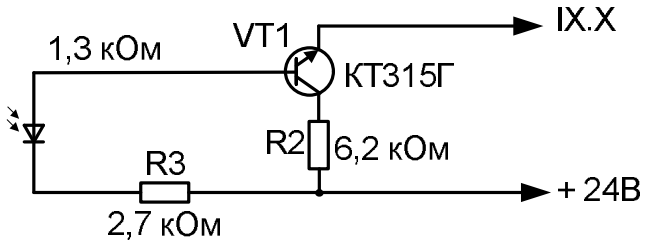


Рис. 3. Блок усиления сигнала оптических датчиков.

Принципиальная схема соединений входных цепей контроллера показана на рис. 5.

На рис. 5 обозначены:

VD14 – VD21 – светодиоды на пульте управления;

R24 –R31 – токоограничивающие сопротивления;

M1 – двигатель главного привода (асинхронный);

M2 – двигатель привода дверей кабины (постоянного тока);

КМ – магнитный пускатель, отключающий главный привод при выходе кабины за пределы рабочей зоны;

K1, K2 – реле управления приводом дверей кабины;

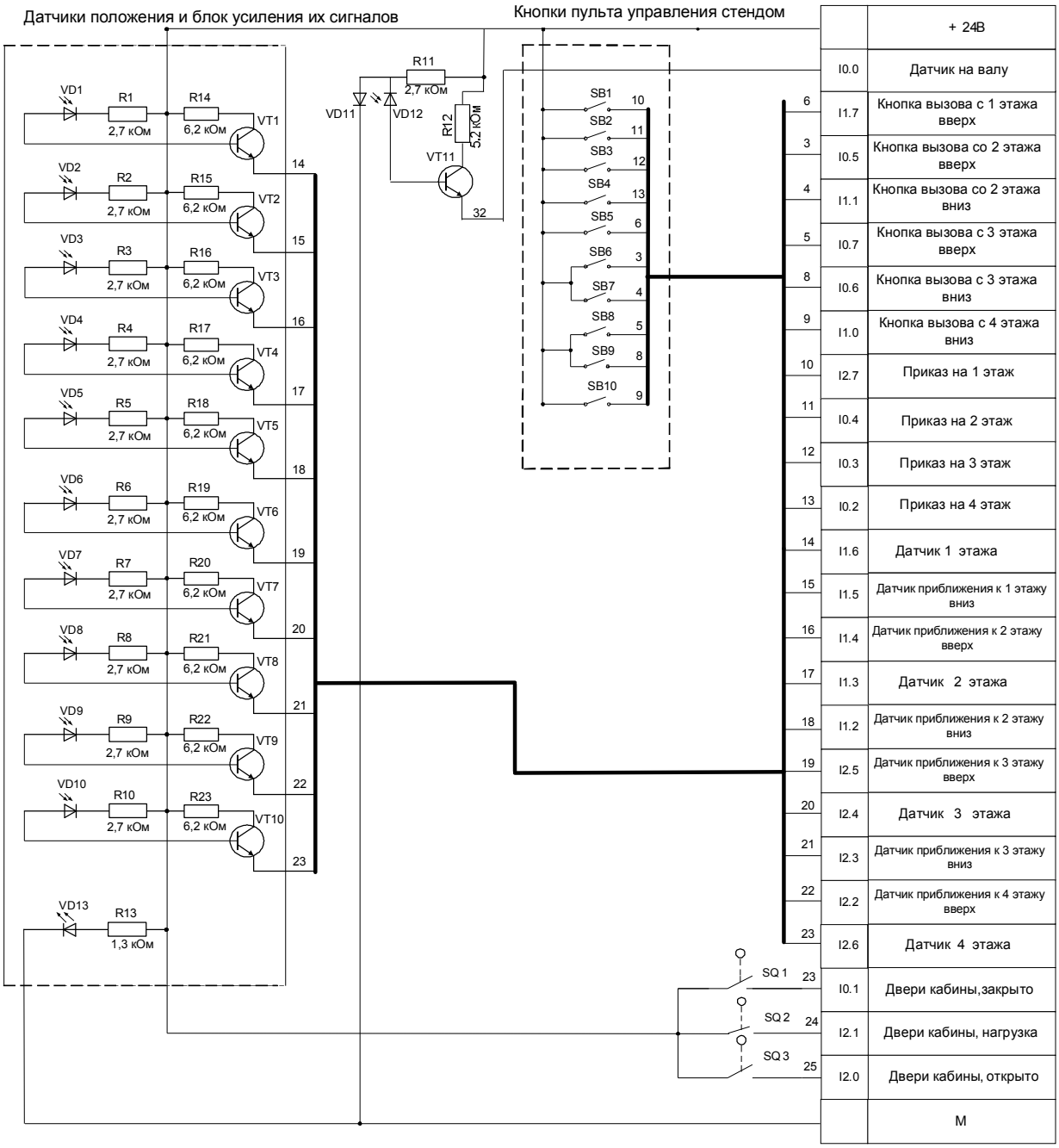


*Рис. 4.*Схема соединений. Входные цепи контроллера.

К3 – реле аварийного отключения главного привода, управляется концевыми выключателями SQ4 и SQ5, расположенными ниже первого этажа и выше четвертого этажа на направляющей шахты.

Выходы контроллера объединены в 3 группы: 1L, 2L, 3L. Под управление преобразователем частоты выделена группа 1L, для управления светодиодами пульта управления и приводом открытия дверей – группы 2L и 3L.

Цепи группы 1L питаются от блока питания преобразователя напряжением 24 В, цепи групп 2L и 3L – от отдельного источника напряжением 12В.

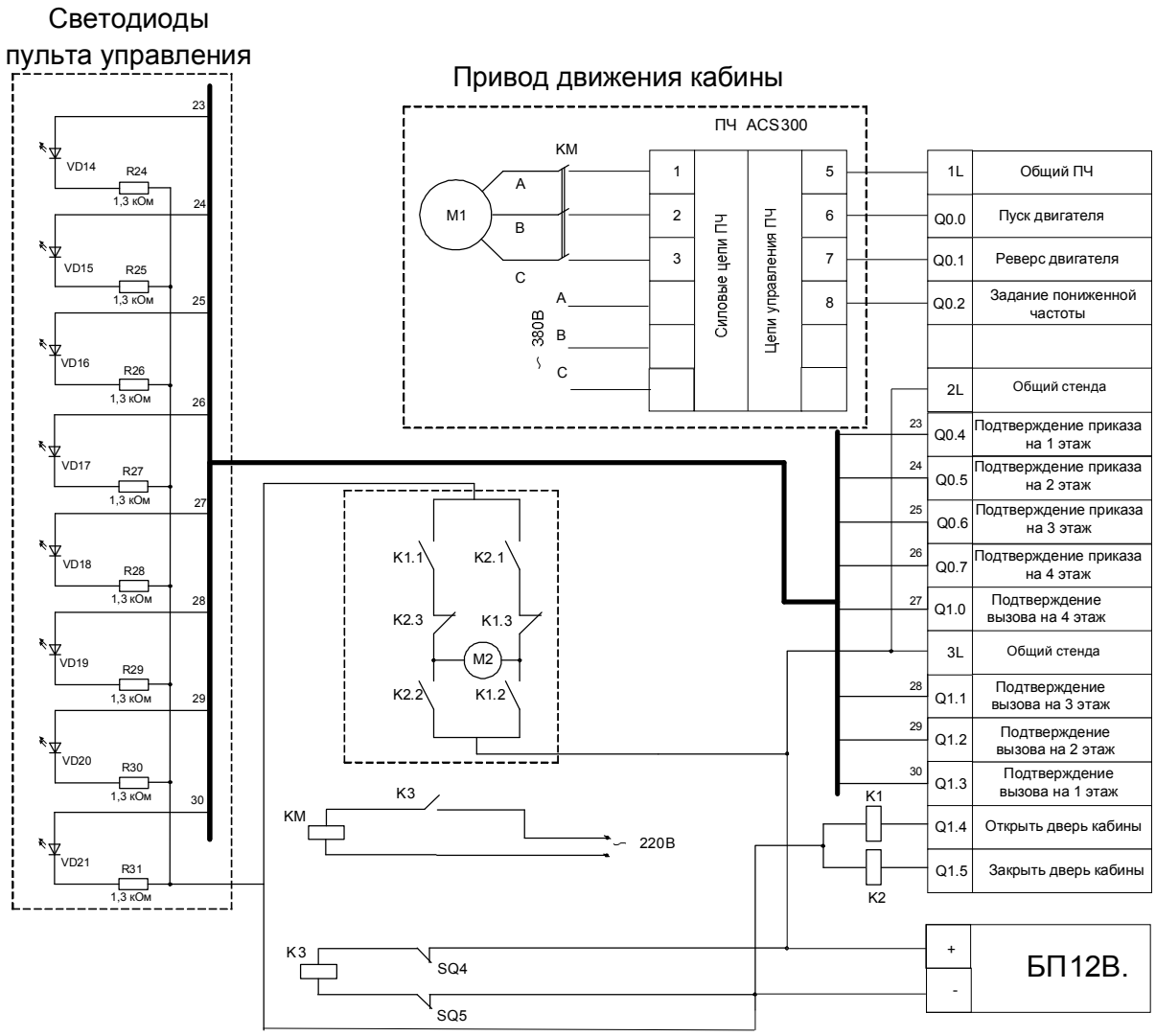


*Рис. 5.*Схема соединений. Выходные цепи контроллера.

Преобразователь частоты

Основные характеристики преобразователя частотыACS 311-2P1-3, входящего в состав главного привода:

входная частота 48 – 63 Гц;

входное напряжение 380 – 400 В;

номинальный входной ток 3,9 А;

выходная частота 0 – 500Гц;

выходное напряжение 0 – $U_{ном}$;

номинальный выходной ток 3,2 А.

Управление ПЧ ACS 300 может осуществляться как непосредственно с панели управления, так и с помощью внешних управляющих сигналов. В первом случае управление называется *локальным* (местным), во втором – *дистанционным*. Просмотр и установка параметров осуществляются с панели управления, приведенной на рис. 6.

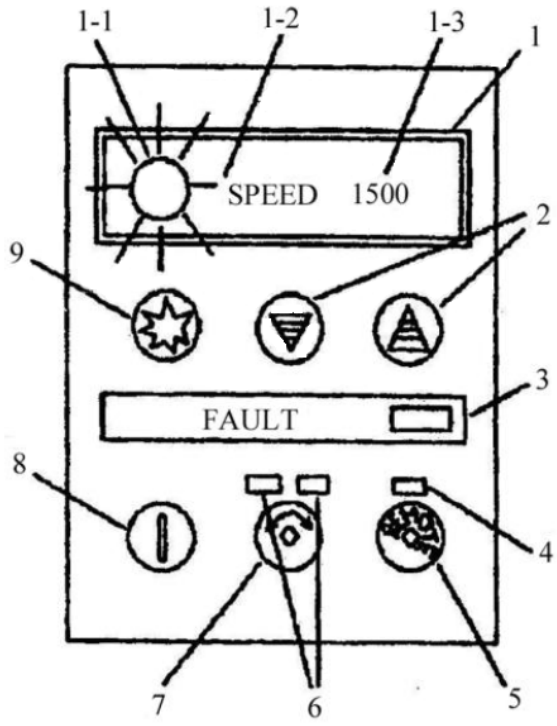


Рис. 6. Панель управления преобразователя.

В состав панели управления входят следующие элементы:

1 – 16-значный алфавитно-цифровой жидкокристаллический дисплей:

1.1 – сегмент дисплея, указывающий на режим работы панели управления (не мигает – режим выбора параметров, мигает – режим установки параметров);

1.2 – часть дисплея, показывающая имя парам;

1.3 – часть дисплея, показывающая значение парам;

2 – кнопки «Увеличение» и «Уменьшение» – используются для просмотра списков параметров и установки значений параметров;

3 – светодиод «Неисправность» – загорается, когда возникает неисправность в электроприводе;

4 – светодиод «Дистанционное управление» – горит, когда ACS 300 находится в режиме управления через блок ввода/вывода (дистанционное управление), не горит, когда управление ACS 300 осуществляется с панели «Локальное управление»;

5 – кнопка «Дистанционное управление» («Remote») – используется для выбора дистанционного или локального управления. Чтобы переключить режим управления, необходимо в течение нескольких секунд держать кнопку нажатой;

6 – светодиод «Направление» – указывает направление вращения двигателя или направление, в котором двигатель будет вращаться после запуска (когда ACS 300 остановлен, светодиоды мигают с небольшой частотой. Если ACS 300 меняет направление вращения, то светодиоды мигают часто);

7 – кнопка «Направление» – позволяет выбрать направление вращения двигателя, светодиоды показывают выбранное направление;

8 – кнопка «Пуск/Стоп» (START/STOP) – включает или останавливает двигатель;

9 – кнопка «Режим» – используется для переключения между режимами выбора и установки параметров.

Информация на дисплей выводится в текстовой форме, а не в виде кода, что облегчает работу в процессе настройки и эксплуатации ПЧ.

При первом подключении к сети ACS 300 по умолчанию переключается в режим дистанционного управления. Аналоговый входной сигнал выбирается при помощи перемычки на плате управления. Это может быть или токовый сигнал в диапазоне от 0(4) до 20 мА, или напряжения в диапазоне от 0(2) до 10 В. Дополнительный переключатель на плате управления используется для конфигурирования дискретных входов и блокировки панели управления.

На рис. 7 приведена схема соединений на плате управления.

Рис. 7. Схема соединений цепей управления.

Реакция ПЧ на сигналы, поступающие на его дискретные входы, определяется положением специального переключателя, расположенного под панелью управления и задающего режим управления ПЧ, а также параметрами программирования. В лабораторных работах используется стандартное распределение («Стандартный режим»), устанавливаемое на заводе изготовителе, согласно которому

вход DI1 отвечает за пуск/останов привода (замыкание контакта на рис. 7 приводит к пуску);

вход DI2 отвечает за реверс, т.е. изменение направления вращения (замыкание контакта на рис. 7 приводит к реверсу);

вход DI3 отвечает за выбор первой фиксированной скорости;

вход DI4 отвечает за выбор второй фиксированной скорости;

вход DI5 отвечает за выбор скорости развертки частоты при разгоне и торможении.

При одновременной активации входов DI3 и DI4 производится выбор третьей фиксированной скорости. Если оба входа неактивны, скорость регулируется потенциометром или входным аналоговым сигналом.

Постоянные скорости программируются с панели управления в виде заданных частот питающего напряжения.

Все программируемые параметры (меню) разделены на четыре страницы. На рис. 8 приведена система меню параметров. Чтобы перейти к другой странице, необходимо нажать кнопку «Режим». Требуемый параметр выбирается кнопками «Увеличение» или «Уменьшение». Чтобы ускорить изменение значения парам, необходимо держать нажатой кнопку «Увеличение» или «Уменьшение».

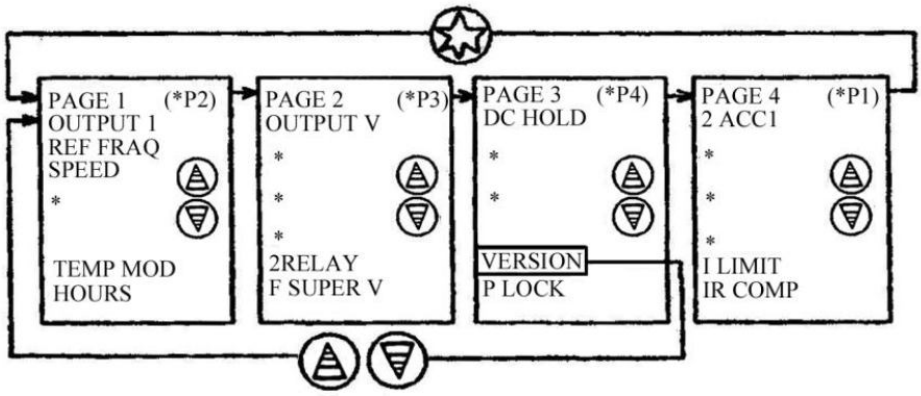


Рис. 8. Система меню параметров.

Ввод программируемых параметров начинается с ввода номинальных (паспортных) параметров двигателя, в состав которых входят номинальная частота вращения двигателя (NOM PRM), номинальная частота питания двигателя (NOM FREQ), номинальное напряжение двигателя (NOM VOLT), номинальный коэффициент мощности двигателя (COS PHI), напряжение питания (сети) (SUPPLY VOLT).

Ниже приведено краткое описание тех параметров, которые, возможно, будут использоваться для просмотра и редактирования при выполнении лабораторных работ.

PAGE 1:

OUTPUT f– частота на выходе ПЧ, подаваемая на двигатель;

SPEED– скорость двигателя, измеряется в оборотах в минуту (об/мин.). Значение, отображаемое на дисплее, действительно только тогда, когда правильно установлен параметр NOM RPM. Скольжение ротора не учитывается. Данные обновляются четыре раза в секунду;

OUTPUT I– рассчитываемый ток фазы двигателя. Погрешность ± 10 %;

MIN FREQ/ MAX FREQ– ввод опорных значений минимальной и максимальной частоты;

ACCTIME 1, DECTIME 1, ACCTIME 2, DECTIME 2 – эти параметры соответствуют промежуткам времени, в течение которых частота выходного напряжения увеличивается от MIN FREQ до MAX FREQ и обратно. Независимо от установок, максимально возможное ускорение/замедление составляет 120

Гц/0,1 с, минимальное – 100 Гц/1800 с. Когда для ввода/вывода выбран стандартный или альтернативный режим, дискретный вход 5осуществляет выбор между ACC/DEC 1 или 2 (0 В – ускорение 1 и + 24 В – ускорение 2);

NOM RPM – номинальная частота вращения, указанная в технических характеристиках двигателя;

NOM FREQ – номинальная частота, указанная в технических характеристиках двигателя (иногда называется точкой ослабления поля). Максимальная выходная частота ACS 300 определяется, исходя из номинальной частоты двигателя: 50 ... 100 Гц – f_{max}= 200 Гц; 101 ... 400 Гц – f_{max}= 500 Гц;

NOM VOLT – номинальное напряжение двигателя (из технических характеристик двигателя). NOM VOLT определяет максимальное выходное напряжение, подаваемое от ACS 300 на двигатель. NOM FREQ задает частоту, при которой напряжение на двигателе равно NOM VOLT. С помощью этих двух параметров ACS 300 можно настроить на работу с конкретным двигателем;

COS PHI– коэффициент мощности (cos φ), выбираемый из технических характеристик двигателя;

SUPPLY VOLT– напряжение сети. Параметр NOM VOLT можно устанавливать только в пределах ± 20 В от значения парам SUPPLY VOLT.

PAGE 2:

OUTPUT V – напряжение, подаваемое на двигатель;

CON f_1, CON f_2, CON f_3 – постоянные частоты (предварительно установленные скорости) 1, 2 и/или 3, которые заменяют опорное значение аналогового входа. Используются в зависимости от значений дискретных входов 3 и 4 или дискретных входов 4 и 5 и выбранного режима управления. Если используется постоянная скорость, то параметры MAX FREQ и MIN FREQ игнорируются;

I LIMIT – этот параметр определяет максимальный выходной ток, подаваемый от ACS 300 к двигателю;

START – способ запуска:

ACC RAMP – постоянное ускорение ACC1 (или ACC2) в зависимости от состояния дискретных входов в стандартном или альтернативном режиме ввода/вывода;

FLYING – используется для запуска двигателя, когда он уже вращается, например, в приводе вентилятора. Привод плавно включится на текущей частоте вращения, вместо того, чтобы начинать с нулевой частоты. Выбор такого значения парам позволит приводу работать в условиях кратковременного отключения сетевого напряжения.

AUTO BOOST – автоматическое увеличение пускового тока, которое может потребоваться для приводов с большим пусковым моментом. Автоматическое увеличение пускового тока действует только при частотах в диапазоне 0 ... 20 Гц или до тех пор, пока не будет достигнуто установленное опорное значение скорости. Увеличение пускового тока не включается, если выходная частота падает ниже 20 Гц при работе двигателя;

FLY+BOOST – одновременно включены функции пуска «На лету» и автоматического увеличения пускового тока;

STOP – способ торможения:

COASTING – после подачи команды «Стоп» ACS 300 отключает напряжение и двигатель вращается по инерции до остановки;

DEC RAMP – постоянное замедление в соответствии с установкой парам DEC1 (или DEC2) в зависимости от состояния дискретных входов в стандартном или альтернативном режимах ввода/вывода;

DC BRAKE – торможение постоянным током (к обмоткам статора прикладывается постоянное напряжение). Используя торможение постоянным током, двигатель можно остановить за наименьшее возможное время без использования резистора торможения;

DEC+BRAKE – используется только тогда, когда подключен тормозной резистор;

DEC+HOLD – постоянное замедление в соответствии с установками. После замедления на период, определяемый параметром DC BRAKE, устанавливается режим DC HOLD;

RAMP – этот параметр позволяет выбрать необходимую форму функций ускорения/замедления:

LINEAR – применяется для приводов, где требуется линейная функция ускорения/замедления;

FAST S – применяется, когда длительность периода ускорения/замедления меньше 1 с;

MEDIUM S – применяется при длительности периода ускорения/замедления меньше 1,5 с;

SLOWS – применяется при длительности периода ускорения/замедления до 15 с.

На рис. 9 приведены графики, демонстрирующие форму ускорения для всех приведенных значений парамRAMP.

SWITCH f – частота широтно-импульсной модуляции. Шум двигателя можно свести к минимуму, установив такую частоту переключения (модуляции) вентилей, которая не создает резонансов в системе. С увеличением частоты переключения эффективность ПЧ падает из-за увеличения коммутационных потерь, что приводит к уменьшению допустимой выходной мощности ПЧ. Но, в то же время, повышается синусоидальность входного тока АД (выходного тока ПЧ), что улучшает его режим работы. Рекомендуется использовать низкую частоту переключения, если система устойчива к шуму.

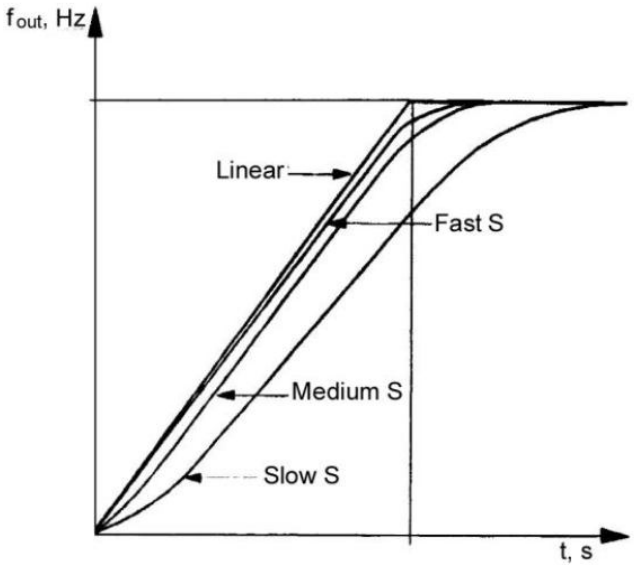


Рис. 9. Формы ускорения.

IR COMP – этот параметр позволяет увеличивать крутящий момент двигателя при скорости вращения от 0,1 Гц до номинальной скорости двигателя. Этот параметр отличается от значения «AUTOBOOST» парам«START» тем, что он всегда действует в диапазоне от 0,1 Гц до номинальной скорости двигателя (рис.10).

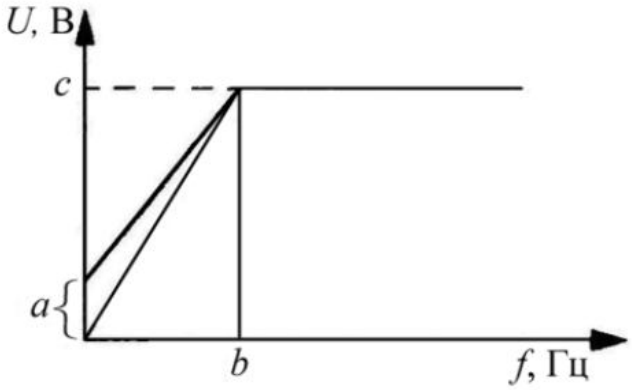


*Рис. 10.*IR-компенсация.

Рекомендуется устанавливать напряжение компенсации как можно меньшим, так как, если уровень компенсации слишком велик, то двигатель может перегреться, или может сработать защита от перегрузки по току.Небольшие двигатели требуют большей компенсации, чем более крупные, так как сопротивление обмоток уменьшается с увеличением мощности двигателя. Если компенсация установлена слишком большой, то двигатель может перейти в режим насыщения, тогда совсем не будет вращаться, потребляя однако при этом ток. Возможные значения парам:

OFF– компенсация не желательна;

0,1 ... 60 В– напряжение компенсации, заданное пользователем;

AUTO– напряжение компенсации устанавливается автоматически для поддержания текущего значения или уменьшения тока;

DC BRAKE – когда значение функции STOP равно DC BRAKE или RAMP+HOLD, данный параметр задает продолжительность подачи на двига-

тель постоянного напряжения в секундах. Если время торможения слишком мало, то по истечении его двигатель будет вращаться по инерции. Подача постоянного напряжения на двигатель нагревает его. При длительном использовании режима DC BRAKE/DEC + HOLD необходимо воспользоваться принудительным охлаждением двигателя.

U/f RATIO – отношение напряжения к частоте (закон скалярного частотного регулирования) в диапазоне частот от 0 Гц до номинальной частоты двигателя может принимать значения LINEAR, SQUARE или OPTIM:

LINEAR – для обеспечения постоянного магнитного потока напряжение на двигателе должно увеличиваться прямо пропорционально (линейно) частоте. Линейное отношение *U/f* (В/Гц) обычно используется в приводах с постоянным нагрузочным моментом;

SQUARE – квадратичная характеристика отношения *U/f* (В/Гц) – обычно используется в приложениях, где момент нагрузки пропорционален квадрату угловой скорости, например, в центробежных нагнетателях (насосах или вентиляторах);

OPTIM – напряжение на двигателе автоматически поддерживается таким, чтобы минимизировать потери в двигателе и уровень шума. Такая установка применяется для приводов с медленно меняющимся нагрузочным моментом и для двигателей, работающих в основном под нагрузкой меньше номинальной.

TEMP LIM – тепловая защита двигателя ACS 300, иногда называемая I^2t, или *твердотельная защита от перегрузки*, включается при помощи парам TEMP LIM. Если значение парам равно OFF, то защита двигателя от перегрузки отключена. Параметры TEMP LIM и MOTOR I определяют область продолжительной безопасной работы двигателя, как это показано на рис. 11.

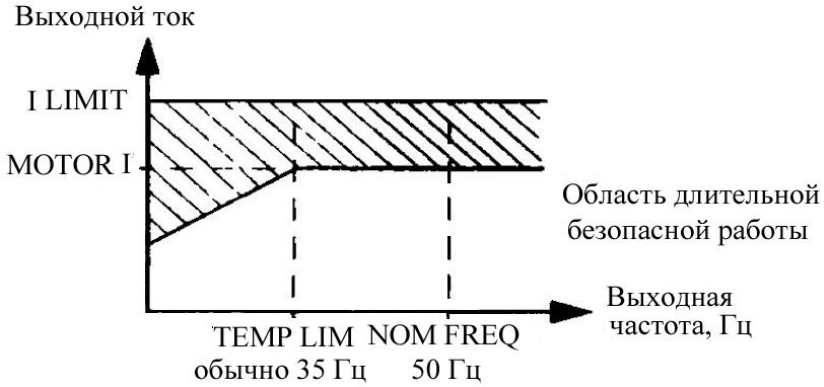


Рис. 11. Тепловая защита двигателя.

Когда ток двигателя превышает уровень, определяемый областью безопасной работы, ACS 300 начинает вычислять прирост температуры двигателя. Когда ACS 300 определит, что температура двигателя превысила допустимые значения, двигатель будет остановлен, сработает реле неисправности и на дисплее появится сообщение о неисправности MOTOR TEMP (температура двигателя).

Сообщение о неисправности можно будет сбросить, когда двигатель достаточно охладится. Если ACS 300 отключить от сети, система тепловой защиты будет сброшена, и после включения предполагается, что двигатель имеет температуру окружающего воздуха.

Функция тепловой защиты рассчитана таким образом, чтобы защитить двигатель даже при малых частотах вращения путем уменьшения допустимого рабочего тока, поскольку на низких частотах вращения охлаждающие вентиляторы двигателя становятся менее эффективными. Для стандартных асинхронных двигателей нормальные условия охлаждения обеспечиваются, начиная с 35 Гц;

MOTOR I – номинальный ток двигателя при полной нагрузке указан на табличке двигателя. Этот параметр не нужно устанавливать, если параметр TEMP LIM имеет значение OFF и не используется значение OPTIM параметра *U/f* RATIO. Параметр MOTOR I никак не ограничивает значение параметра ILIMIT.

PAGE 3:

DC HOLD – этот параметр предусматривает использование функции DC HOLD (стоянка под током). Параметр может принимать три значения:

0 (выключено) – функция DC HOLD отключена;

1 (нормальный) – при выборе этого значения на двигатель подаются «меньший» постоянный ток и уровень момента вращения, чем в случае выбора значения «2». Рекомендуется сначала использовать это значение парам, чтобы проверить, обеспечивается ли достаточная степень удержания;

2 (сильный) – при выборе этого значения на двигатель подаются требуемый постоянный ток и уровень момента вращения.

Когда опорная и выходная частота опускаются ниже 1,5 Гц, ACS 300 перестает генерировать синусоидальный ток и подает на двигатель постоянный ток. Когда опорная частота поднимается выше 1,5 Гц, подача постоянного тока прекращается и возобновляется нормальная работа ACS 300.

Функция DC HOLD не работает, если сигнал START отсутствует.

ПЛК Siemens S7 200

Описание и характеристики

Программируемый логический контроллер Siemens S7 226, управляющий лабораторной установкой, входит в состав семейства SIMATICS7-200. Семейство объединяет в своем составе модули центральных процессоров, коммуникационные модули, модуль позиционирования EM 253, модули ввода-вывода дискретных и аналоговых сигналов, модули блоков питания. Все модули способны работать в диапазоне температур от 0 до +55°C.

Конструктивные особенности:

компактные пластиковые корпуса со степенью защиты IP20;

простое подключение внешних цепей через терминальные блоки с контактами под винт. Защита всех токоведущих частей открывающимися пластиковыми крышками;

наличие штатных или опциональных съемных терминальных блоков, позволяющих выполнять замену модулей без демонтажа их внешних цепей;

монтаж на стандартную 35мм профильную шину или на плоскую поверхность с креплением винтами;

соединение модулей с помощью плоских кабелей, вмонтированных в каждый модуль расширения.

Основные характеристики Siemens S7 226 приведены в табл. 1.

Таблица 1

Характеристики Siemens S7 226

Параметр	Значение
Физические размеры(мм)	190 x 80 x 62
Программная память: с редактированием в режиме RUN без редактирования в режиме RUN	 16384 байта 24576 байт
Память данных	10240 байт
Буферизация памяти без/с буферной батареей	100 часов/200 дней
Локальные встроенные входы/выходы дискретные аналоговые	 24 вх./16 вых. -
Модули расширения	до 7 модулей
Скоростные счетчики 1-фазные 2-фазные	 6 при 30 кГц 4 при 20 кГц
Импульсные выходы(DC)	2 при 20 кГц
Часы реального времени	Встроенные
Коммуникационные порты	2 RS-485
Арифметика с плавающей точкой	Поддерживается
Дискретные входы/выходы (образ процесса)	256 (128 входов, 128 выходов)
Времена выполнения булевых операций	0,22 микросекунд/операцию
Выходной ток встроенного блока питания	400 мА

Встроенный интерфейс RS 485 (один или два) используется:

без дополнительного программного обеспечения:

для программирования контроллера;

для включения контроллера в сети PPI или MPI со скоростью передачи данных до 187.5 Кбит/с;

в качестве свободно программируемого порта с поддержкой ASCII протокола и скоростью передачи данных до 38.4 Кбит/с;

с дополнительным программным обеспечением Instruction Library:

для поддержки протокола USS со скоростью передачи данных до 19.2 Кбит/с и возможностью подключения до 30 преобразователей частоты (например, преобразователей серий MICROMASTER или SINAMICS);

для поддержки протокола MODBUS RTU и работы в режиме ведомого сетевого устройства.

Центральный процессор оснащен встроенным блоком питания 24В для питания датчиков или другой нагрузки.

Дискретные входы рассчитаны на входное напряжение 24В.

Для программирования контроллера используется пакет STEP 7-Micro/WIN. Основным элементом программы, загружаемой в контроллер, является т.н. *программный блок*.

Программный блок состоит из исполняемого кода и комментариев. Исполняемый код состоит из основной программы, а также подпрограмм и программ обработки прерываний. Код компилируется и загружается в S7–200. Комментарии не компилируются и не загружаются.

Основная программа содержит команды, управляющие приложением. Контроллер выполняет эти команды последовательно и однократно в каждом цикле.

Подпрограммы выполняются только тогда, когда они вызываются: основной программой, программой обработки прерываний или другой подпрограммой. Подпрограммы полезны, если какая-нибудь функция выполняется многократно. Чтобы не переписывать логику в каждом месте основной программы, можно записать логику функции один раз в подпрограмме, а затем вызывать эту подпрограмму столько раз, сколько необходимо при выполнении основной программы. Подпрограммы имеют много преимуществ:

использование подпрограмм уменьшает общую величину программы;

использование подпрограмм уменьшает время цикла. S7–200 в каждом цикле анализирует код в основной программе независимо от того, исполняется этот код или нет, но код в подпрограмме анализируется только тогда, когда подпрограмма вызывается, и не анализируется в циклах, в которых подпрограмма не вызывается;

с помощью подпрограмм создается мобильный код. Подпрограммы легко переносятся, если используют локальные переменные.

Программы обработки прерываний реагируют на определенные прерывающие события. Программа обработки прерываний проектируется для обработки заранее определенных прерывающих событий. S7–200 исполняет программу обработки прерываний, когда возникает соответствующее событие. Программы обработки прерываний не вызываются основной программой.

STEP 7-Micro/WIN имеет в своем распоряжении три редактора для создания программ: цепная логическая схема (LAD), называемая также контактным планом (KOP), список команд (STL или AWL) и функциональная блок-схема (FBD), называемая также функциональным планом (FUP). С некоторыми ограничениями программы, написанные в любом из этих редакторов программ, могут отображаться и редактироваться с помощью других редакторов программ. При выполнении лабораторных работ рекомендуется использовать язык LAD.

Доступ к данным

S7–200 хранит информацию в различных местах памяти, которые имеют однозначные адреса. Требуется явно указать адрес в памяти, к которому вы хотите обратиться.

Таблица 2

Форматы данных

Представление	Байт (B)	Слово (W)	Двойное слово (D)
Целое без знака	от 0 до 255 от 0 до FF	от 0 до 65 535 от 0 до FFFF	от 0 до 4 294 967 295 от 0 до FFFF FFFF
Целое со знаком	от -128 до +127 от 80 до 7F	от –32 768 до +32 767 от 8000 до 7FFF	от –2 147 483 648 до +2 147 483 647 от 8000 0000 до 7FFF FFFF
Вещественное IEEE 32–битовое с плавающей точкой	*Неприменимо*	*Неприменимо*	от +1.175495E-38 до +3.402823E+38 (положительное) от –1.175495E-38 до –3.402823E+38 (отрицательное)

Для обращения к биту в некоторой области памяти нужно указать адрес бита. Этот адрес состоит из идентификатора области памяти, адреса байта и номера бита. На рис. 12 показан пример обращения к биту (адресация в формате «байт.бит»). В этом примере за областью памяти и адресом байта (I = input [вход], 3 = байт 3) следует точка («.»), чтобы отделить адрес бита (бит 4).

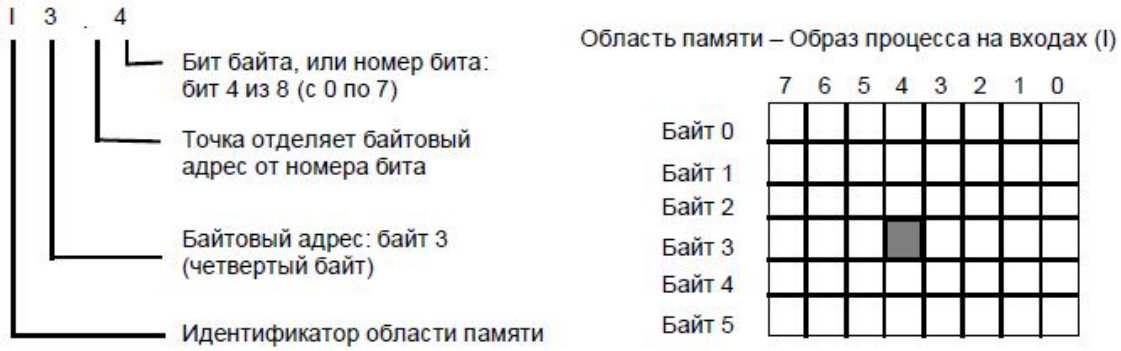


Рис. 12. Адресация битов.

Применяя формат байт.бит, можно обратиться к данным в большинстве областей памяти (V, I, Q, M, S, L и SM) как к байтам, словам или двойным словам. Если требуется обратиться к целому байту, слову или двойному слову данных в памяти, нужно указать эти адреса подобно адресу бита. Указываются идентификатор области, обозначение длины данных и начальный адрес байта, слова или двойного слова, как показано на рис. 13.

К данным в других областях памяти (напр., T, C, HC и аккумуляторы) обращаются, указывая в качестве адреса идентификатор области и номер элемента.

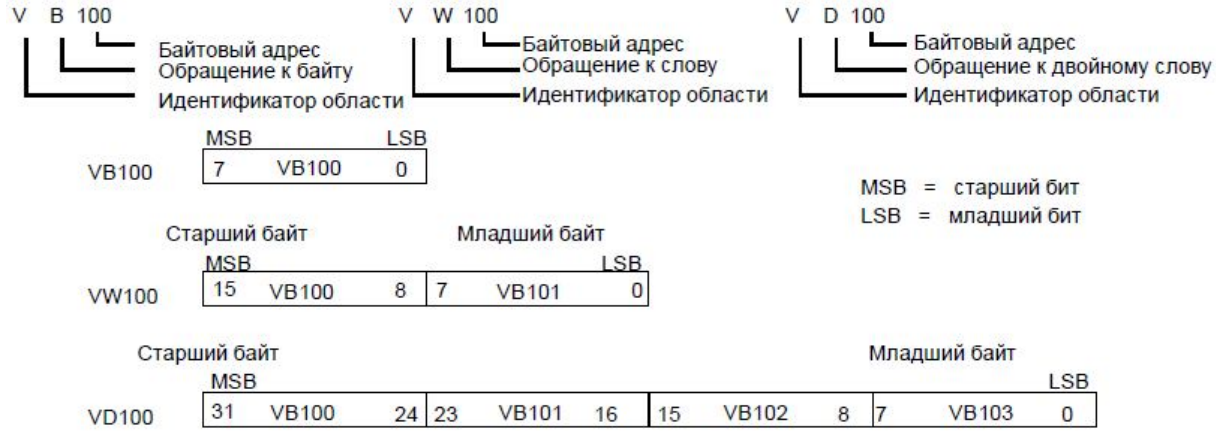


Рис. 13. Обращение к одному и тому же адресу в формате байта, слова и двойного слова.

Обращение к данным в областях памяти

Регистр входов образа процесса: I

В начале каждого цикла S7–200 опрашивает физические входы и записывает полученные значения в регистр входов образа процесса. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Бит: I*[адрес байта].[адрес бита]* I0.1

Байт, слово или двойное слово: I*[длина][начальный адрес байта]* IB4

Регистр выходов образа процесса: Q

В конце цикла S7–200 копирует значения, хранящиеся в регистре выходов образа процесса, в физические выходы. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	Q*[адрес байта].[адрес бита]*	Q1.1
Байт, слово или двойное слово:	Q*[длина][начальный адрес байта]*	QB5

Область памяти переменных: V

Память переменных можно использовать для хранения промежуточных результатов операций, выполняемых в программе. В памяти переменных можно хранить также другие данные, имеющие отношение к процессу или к решению вашей задачи автоматизации. К памяти переменных можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	V*[адрес байта].[адрес бита]*	V10.2
Байт, слово или двойное слово:	V*[длина][начальный адрес байта]*	VW100

Область битовой памяти: M

Биты памяти (меркеры) можно использовать как управляющие реле для хранения промежуточных результатов операций или другой управляющей информации. К битам памяти можно обратиться в формате бита, байта, слова и двойного слова:

Бит:	M*[адрес байта].[адрес бита]*	M26.7
Байт, слово или двойное слово:	M*[длина][начальный адрес байта]*	MD20

Таймеры: T

S7–200 имеет в своем распоряжении таймеры, которые отсчитывают приращения времени с разрешениями (шагами базы времени) 1 мс, 10 мс или 100 мс. С таймером связаны две переменные:

текущее значение: это 16-битовое целое со знаком хранит количество времени, отсчитанное таймером;

бит таймера: этот бит устанавливается или сбрасывается, когда текущее значение становится равным предустановленному значению. Предустановленное значение вводится как часть таймерной команды.

К обоим этим элементам данных обращаются через адрес таймера (T + номер таймера). Происходит ли обращение к биту таймера или к текущему значению, зависит от используемой команды: команды с операндами в битовом формате обращаются к биту таймера, тогда как команды с операндами в формате слова обращаются к текущему значению. Как показано на рис. 14, команда «Нормально открытый контакт» обращается к биту таймера, а команда «Передать слово» обращается к текущему значению таймера.

Рис. 14. Обращение к биту или к текущему значению таймера.

Счетчики: C

S7–200 имеет в своем распоряжении три вида счетчиков, которые подсчитывают нарастающие фронты на счетных входах счетчика: один вид счетчиков ведет прямой счет, другой считает только в обратном направлении, а третий вид считает в обоих направлениях. Со счетчиком связаны две переменные:

текущее значение: это 16-битовое целое со знаком хранит счетное значение, накопленное счетчиком;

бит счетчика: этот бит устанавливается или сбрасывается, когда текущее значение становится равным предустановленному значению. Предустановленное значение вводится как часть команды счетчика.

К обоим этим элементам данных обращаются через адрес счетчика (C + номер счетчика). Происходит ли обращение к биту счетчика или к текущему значению, зависит от используемой команды: команды с операндами в битовом формате обращаются к биту счетчика, тогда как команды с операндами в формате слова обращаются к текущему значению. Как показано на рис. 15, команда «Нормально открытый контакт» обращается к биту счетчика, а команда «Передать слово» обращается к текущему значению счетчика.

Рис. 15. Обращение к биту или к текущему значению счетчика.

Скоростные счетчики: HC

Скоростные счетчики подсчитывают быстрые события независимо от цикла CPU. Скоростные счетчики имеют в своем распоряжении 32-битовое целое счетное значение(текущее значение). Для обращения к счетному значению скоростного счетчика нужно ввести его адрес, указав область памяти (HC) и номер счетчика (например, HC0). Текущее значение скоростного счетчика защищено от записи и может быть адресовано только в формате двойного слова (32 бита).

Формат: HC*[номер скоростного счетчика]* HC1

Аккумуляторы: AC

Аккумуляторы – это элементы чтения/записи, которые могут использоваться как память. Можно использовать аккумуляторы для передачи параметров в подпрограммы и из них или для хранения промежуточных результатов расчетов. S7–200 имеет в своем распоряжении четыре 32-битовых аккумулятора (AC0, AC1, AC2 и AC3).

К данным в аккумуляторах можно обратиться в формате бита, слова или двойного слова.

Длина данных, к которым производится обращение, зависит от команды, которая используется для обращения к аккумулятору. Как показано на рис. 16, при обращении к аккумулятору в формате бита или слова используются младшие 8 или 16 битов значения, хранящегося в аккумуляторе. При обращении к аккумулятору в формате двойного слова используются все 32 бита.

Формат: AC*[номер аккумулятора]* AC0

Специальные биты памяти: SM

Специальные биты памяти (SM) предоставляют средство для обмена данными между CPU и вашей программой. Можно использовать эти биты для выбора и управления некоторыми специальными функциями CPU S7–200, например, бит, который устанавливается только в первом цикле, бит, который устанавливается и сбрасывается с фиксированной частотой, или бит, который указывает на состояние арифметической или иной команды.

К SM-битам можно обращаться в формате бита, слова или двойного слова:

Бит:	SM*[адрес байта].[адрес бита]*	SM0.1
Байт, слово или двойное слово:	SM*[длина][начальный адрес байта]*	SMB86

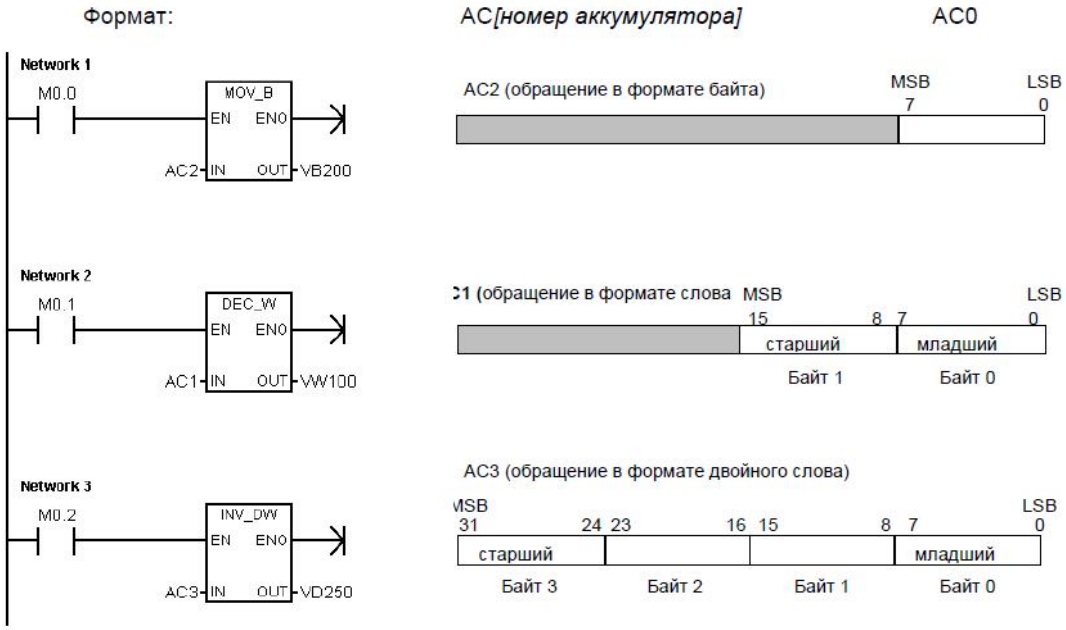


Рис. 16. Обращение к аккумуляторам.

Память локальных данных: L

S7–200 имеет в своем распоряжении 64 байта локальной памяти, из которых 60 могут быть использованы в качестве промежуточной памяти или для передачи формальных параметров в подпрограммы.

Память локальных данных похожа на память переменных с одним существенным отличием. Память переменных доступна глобально, тогда как память локальных данных доступна локально. Глобальная доступность означает, что к адресу в этой области памяти можно обратиться из любой организационной единицы программы (из основной программы, подпрограммы или подпрограмм обработки прерываний). Локальная доступность означает, что эта область памяти ставится в соответствие определенной организационной единице программы. S7–200 выделяет 64 байта локальной памяти для главной программы, 64 байта для каждого уровня вложенности подпрограмм и 64 байта для программ обработки прерываний.

К области локальных данных, поставленной в соответствие основной программе, не имеют доступа подпрограмм и программы обработки прерываний. Подпрограмма не может обращаться к области локальных данных основной программы, программы обработки прерываний или другой подпрограммы. Аналогично, программа обработки прерываний не имеет доступа к области локальных данных основной программы или подпрограммы.

S7–200 выделяет область локальных данных по мере необходимости. Это значит, что при выполнении основной программы области локальных данных для подпрограмм и программ обработки прерываний не существуют. Если возникает прерывание или вызывается подпрограмма, то по потребности выделяется локальная память. Вновь выделенная локальная память может снова использовать те же адреса, которые использовались другой подпрограммой или программой обработки прерываний.

S7–200 не инициализирует область локальных данных к моменту ее назначения, поэтому она может содержать любые значения. Если при вызове под-

программы передаются формальные параметры, то S7–200 сохраняет значения передаваемых параметров в соответствующих адресах области локальных данных, выделенной этой подпрограмме.

Адреса в области локальных данных, которые не получили значений при передаче формальных параметров, не инициализируются и при выделении могут содержать произвольные значения.

Бит:	L*[адрес байта].[адрес бита]*	L0.0
Байт, слово или двойное слово:	L*[длина] [начальный адрес байта]*	LB33

Вещественные числа

Вещественные числа (или числа с плавающей точкой) представляются как 32-битовые числа однократной точности, формат которых описан в стандарте ANSI/IEEE 754-1985. Обращение к вещественным числам производится в формате двойного слова. У S7–200 числа с плавающей точкой имеют точность до 6 десятичных разрядов. Поэтому при вводе константы с плавающей точкой можно указывать до 6 десятичных разрядов.

Расчеты, включающие длинные последовательности значений, содержащие очень большие и очень малые числа, могут привести к неточным результатам. Это может произойти, если числа отличаются друг от друга в 10 в степени x раз, где $x > 6$. Например: 100 000 000 + 1 = 100 000 000.

Косвенная адресация областей памяти S7–200 с помощью указателей

Косвенная адресация использует указатель для доступа к данным в памяти. Указатели – это ячейки памяти, имеющие размер двойного слова, которые содержат адрес другой ячейки памяти. В качестве указателей можно использовать только ячейки памяти переменных и локальных данных или аккумуляторные регистры (AC1, AC2 или AC3). Для создания указателя необходимо использовать команду «Переместить двойное слово». Эта команда передает адрес косвенно адресованной ячейки памяти в ячейку указателя.

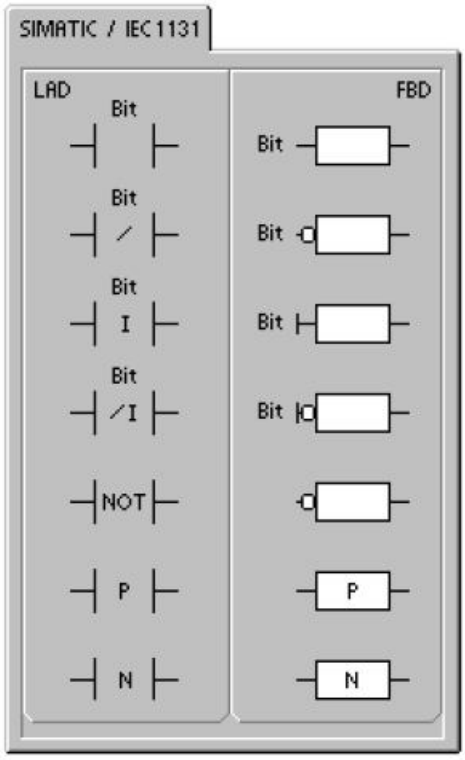
Указатели могут также передаваться в подпрограмму в качестве параметров.

S7–200 дает возможность использования указателей для косвенной адресации следующих областей памяти: I, Q, V, M, S, AI, AQ, SM, T (только текущее значение) и C (только текущее значение). Косвенную адресацию нельзя использовать для обращения к отдельному биту или к областям памяти HC или L. Если вы хотите косвенно обратиться к данным, расположенным по некоторому адресу в памяти, вы можете создать указатель на этот адрес, введя амперсанд (&) и соответствующий адрес. Входному операнду команды должен предшествовать амперсанд(&), чтобы указать на необходимость перемещения в ячейку, обозначенную в выходном операнде команды (указателе), адреса ячейки памяти, а не ее содержимого.

Ввод астериска (*) перед операндом команды указывает, что этот операнд является указателем.

Команды

Стандартные контакты



Команды«Нормально открытый контакт» (LD, A и O) и*«Нормально замкнутый контакт»* (LDN, AN, ON) получают исходное значение из памяти или из регистра образа процесса (или из регистра образа процесса, если типом данных является I или Q).

Нормально открытый контакт замкнут (включен), когда бит равен 1, а нормально замкнутый контакт замкнут(включен), когда бит равен 0. В FBD к блокам И и ИЛИ может быть подключено не более 32 входов.

Непосредственно управляемый контакт при своей актуализации не зависит от цикла S7–200, его значение обновляется немедленно. Команды «Непосредственно управляемый нормально открытый контакт» (LDI, AI и OI) и «Непосредственно управляемый нормально замкнутый контакт» (LDNI, ANI и ONI) при выполнении команды получают значение физического входа, однако, регистр образа процесса не обновляется.

Непосредственно управляемый нормально открытый контакт замкнут (включен), когда физический вход (бит)находится в состоянии 1, а непосредственно управляемый нормально замкнутый контакт замкнут(включен), когда физический вход (бит) находится в состоянии 0. Команды, представляющие непосредственно управляемый нормально открытый контакт, непосредственно загружают значение физического входа в вершину стека или выполняют логическое сопряжение значения физического входа со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ, а команды, представляющие непосредственно управляемый нормально замкнутый контакт, непосредственно загружают логическое отрицание значения физического входа в вершину стека или выполняют логическое сопряжение отрицания значения физического входа со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ.

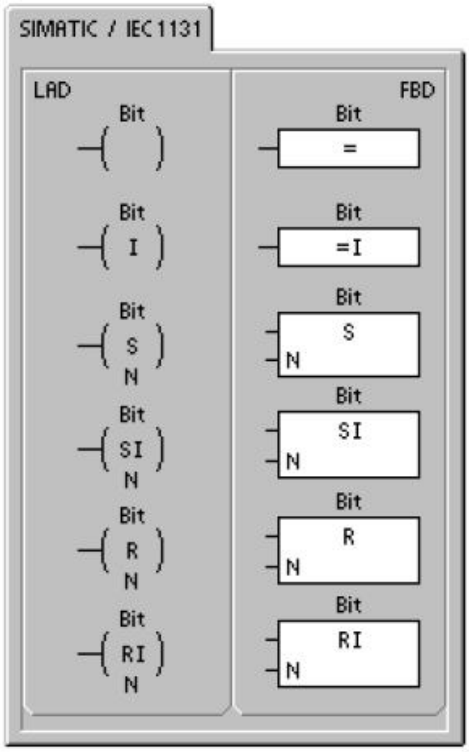
Команда отрицания (NOT) изменяет состояние входа потока сигнала (т.е. она изменяет значение в вершине стека с 0 на 1 или с 1 на 0).

Контакт «Положительный фронт» (EU) пропускает поток сигнала в течение одного цикла при каждом появлении положительного фронта.

Контакт «Отрицательный фронт» (ED)пропускает поток сигнала в течение одного цикла при каждом появлении отрицательного фронта. У команды «Положительный фронт» при обнаружении перехода значения в вершине стека с 0 на 1 значение в вершине стека устанавливается в 1; в противном случае оно устанавливается в 0. У команды «Отрицательный фронт» при обнаружении пе-

рехода значения в вершине стека с 1 на 0 значение в вершине стека устанавливается в 1; в противном случае оно устанавливается в 0.

Катушки



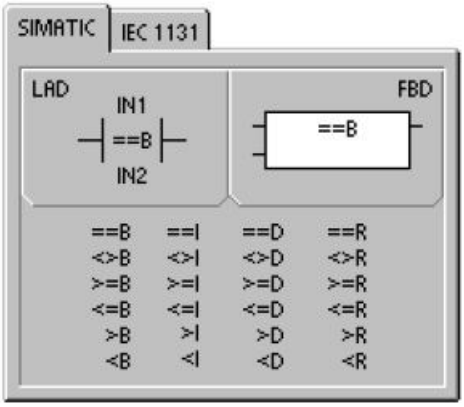
Команда присваивания (=) записывает новое значение для выходного бита в регистр образа процесса. При выполнении команды присваивания S7–200устанавливает или сбрасывает выходной бит в регистре образа процесса. В LAD и FBD указанный бит устанавливается равным потоку сигнала.

Команда непосредственного присваивания битового значения (=I) при своем выполнении записывает новое значение как в физический выход, так и в образ процесса. Когда выполняется команда непосредственного присваивания битового значения, физический выход(бит) немедленно устанавливается в соответствии с состоянием потока сигнала.

Команды установки (S) и *сброса* (R) устанавливают(включают) или сбрасывают (выключают) указанное количество входов или выходов (N), начиная с указанного адреса (бита). Можно установить или сбросить от 1 до 255 входов и выходов. Если команда сброса указывает на бит таймера (T) или счетчика (C), то команда сбрасывает бит таймера или счетчика и стирает текущее значение таймера или счетчика.

Команды непосредственной установки и *непосредственного сброса* непосредственно устанавливают (включают) или непосредственно сбрасывают (выключают) указанное количество входов или выходов (N), начиная с указанного адреса (бита). Можно непосредственно и немедленно установить или сбросить от 1 до 128 входов и выходов.

Сравнение числовых величин



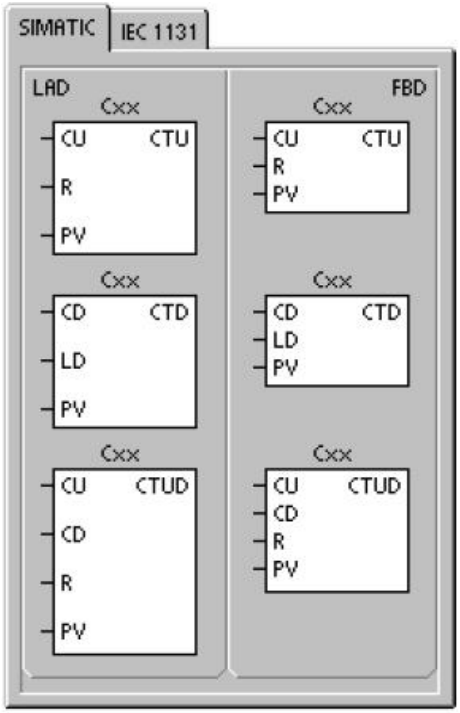
Команды сравнения используются для сравнения двух величин:

IN1 = IN2 IN1 >= IN2 IN1 <= IN2

IN1 > IN2 IN1 < IN2 IN1 <> IN2

Операции сравнения байтов не учитывают знака. Операции сравнения целых чисел, в том числе слов и двойных слов, учитывают знак. Операции сравнения вещественных чисел также учитывают знак.

Счетчики

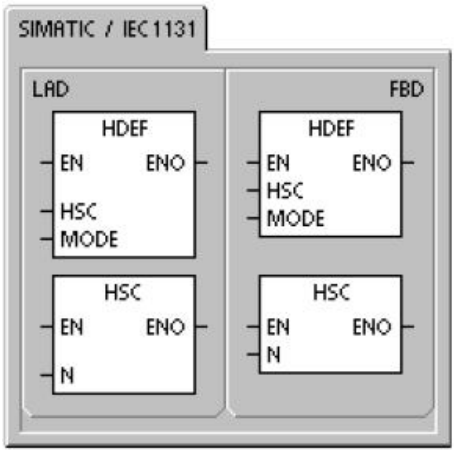


Команда прямого счета (CTU) увеличивает текущее значение счетчика при появлении нарастающего фронта на входе (CU). Когда текущее значение Cxx больше или равно предустановленному значению PV, бит счетчика Cxx устанавливается. Счетчик сбрасывается, когда включается вход сброса (R), или когда выполняется команда сброса. Счетчик прекращает счет при достижении максимального значения (32767).

Команда обратного счета (CTD) уменьшает текущее значение счетчика при появлении нарастающего фронта на входе (CD). Когда текущее значение Cxx равно нулю, бит счетчика Cxx включается. Счетчик сбрасывает свой бит Cxx и загружает текущее значение предустановленным значением PV, когда включается вход загрузки LD. Счетчик останавливается, когда он достигает нуля, и бит счетчика Cxx включается.

*Команда реверсивного счета*CTUD имеет два счетных входа CU и CD, вход сброса Ri использует предустановленное значение PV.

Скоростные счетчики



Команда определения режима работы скоростного счетчика (HDEF) устанавливает режим работы для определенного скоростного счетчика (HSCx). Выбором режима определяются датчик тактовых импульсов, направление и функции запуска и сброса скоростного счетчика.

Команда активизации скоростного счетчика (HSC) конфигурирует и управляет режимом работы скоростного счетчика через сигнальные состояния битов специальной памяти HSC. Параметр N определяет номер скоростного счетчика.

Скоростные счетчики могут быть сконфигурированы на двенадцать различных режимов работы (табл. 3).

Каждый счетчик имеет специализированные входы, которые поддерживают такие функции, как датчик тактовых импульсов, управление направлением, сброс и запуск. Для двухфазных счетчиков оба датчика тактовых импульсов могут работать со своей максимальной скоростью. В квадратурных режимах (A/B-счетчики) предоставляется возможность выбора однократной (1x) или четырехкратной (4x) скорости счета. Все счетчики работают с максимальной скоростью, не создавая помех друг другу.

CPU 226 поддерживают шесть скоростных счетчиков: от HSC0 до HSC5.

В большинстве случаев применения скоростной счетчик загружается первым из нескольких предустановленных значений, и желаемые выходы активизируются на интервал времени, в течение которого текущее значение счетчика меньше текущего предустановленного значения. Счетчик настроен таким образом, что, когда текущее значение счетчика становится равным предустановленному значению, или при появлении сброса происходит прерывание.

Когда при равенстве текущего значения счетчика и предустановленного значения происходит прерывающее событие, загружается новое предустановленное значение, и устанавливается следующее состояние для выходов. Когда происходит событие, вызывающее прерывание по сбросу, то устанавливаются первое предустановленное значение и первые состояния выходов, и цикл повторяется.

Все счетчики в одном и том же режиме работают одинаково. Имеется четыре основных вида счетчиков: однофазный счетчик с внутренним управлением направлением, однофазный счетчик с внешним управлением направлением, двухфазный счетчик с 2 тактовыми входами и квадратурный счетчик с фазами A и B. Не каждый счетчик поддерживает все режимы. Каждый счетчик можно использовать: без входов сброса и пуска, со сбросом, но без пуска, или с входами пуска и сброса.

Перед использованием скоростного счетчика необходимо с помощью команды HDEF (HighSpeed Counter Definition – определение скоростного счетчика) выбрать его режим. С помощью бита памяти первого цикла SM0.1 (этот бит включен в течение первого цикла обработки программы, а затем выключается) вызывается фрагмент кода, который содержит команду HDEF.

Для программирования скоростного счетчика требуется выполнить следующие основные задачи:

1) определить счетчик и режим;
2) настроить управляющий байт;
3) установить текущее (начальное) значение;
4) задать предустановленное (целевое) значение;
5) назначить и разблокировать программу обработки прерываний;
6) активизировать скоростной счетчик.

Режимы работы скоростных счетчиков приведены в табл. 3.

Таблица 3

Режимы работы скоростных счетчиков

Режим	Описание	Входы			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.6	I0.7	I1.0	I1.1
	HSC2	I1.2	I1.3	I1.4	I1.5
	HSC3	I0.1			
	HSC4	I0.3	I0.4	I0.5	
	HSC5	I0.4			
0	Однофазный счетчик с внутренним управлением направлением	Датчик тактовых импульсов			
1		Датчик тактовых импульсов		Сброс	
2		Датчик тактовых импульсов		Сброс	Пуск
3	Однофазный счетчик с внешним управлением направлением	Датчик тактовых импульсов	Направление		
4		Датчик тактовых импульсов	Направление	Сброс	
5		Датчик тактовых импульсов	Направление	Сброс	Пуск
6	Двухфазный счетчик с 2 тактовыми входами	Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления		
7		Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления	Сброс	
8		Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления	Сброс	Пуск
9	Квадратурный счетчик с фазами A и B	Датчик тактовых импульсов A	Датчик тактовых импульсов B		
10		Датчик тактовых импульсов A	Датчик тактовых импульсов B	Сброс	
11		Датчик тактовых импульсов A	Датчик тактовых импульсов B	Сброс	Пуск
12	Режим счета 12 поддерживают только HSC0 и HSC3. HSC0 считает количество импульсов, выдаваемых Q0.0. HSC3 считает количество импульсов, выдаваемых Q0.1.				

Временные диаграммы на рис.17-21 показывают, как работает каждый счетчик в соответствии с режимом.

Рис. 17. Пример работы в режимах 0, 1 или 2.

Рис. 18. Пример работы в режимах 3, 4 или 5.

Когда используются режимы счета 6, 7 или 8, и в течение 0,3 микросекунды друг за другом появляется нарастающий фронт на тактовых входах счета вперед и счета назад, скоростной счетчик может рассматривать эти события как происходящие одновременно.

Если это происходит, то текущее значение не меняется и не отображается изменение в направлении счета. Если между поступлениями нарастающих фронтов на тактовые входы счета вперед и счета назад проходит больше 0,3 микросекунды, то скоростной счетчик воспринимает эти события отдельно. В этом случае ошибки не происходит, и счетчик сохраняет правильное счетное значение.

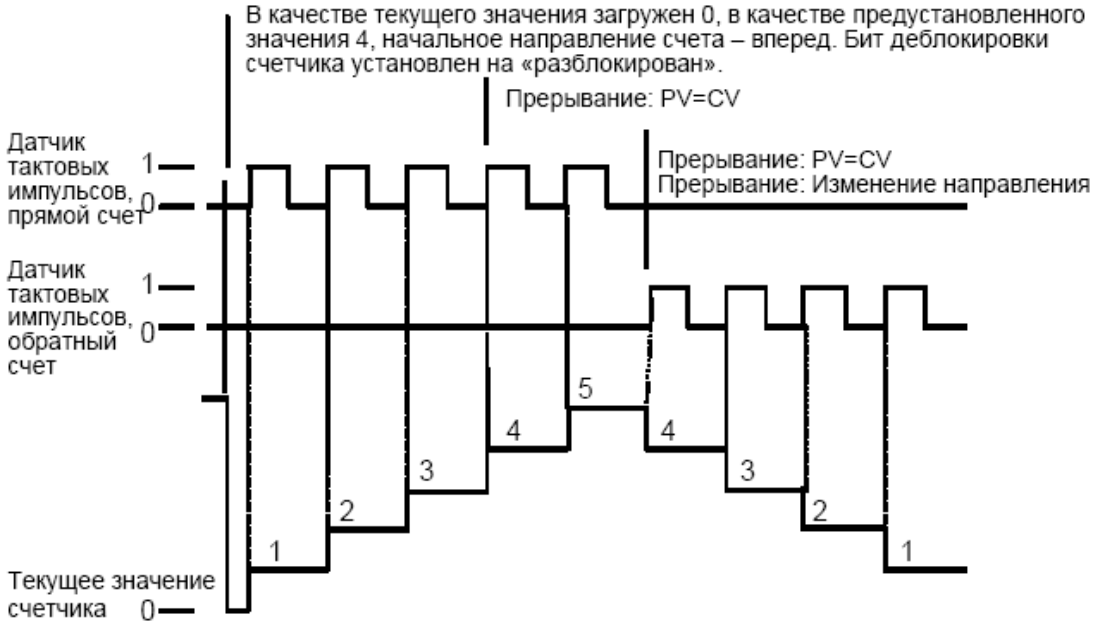


Рис. 19. Пример работы в режимах 6, 7 или 8.

Рис. 20. Пример работы в режимах 9, 10 или 11 (квадратурный режим, однократная скорость).

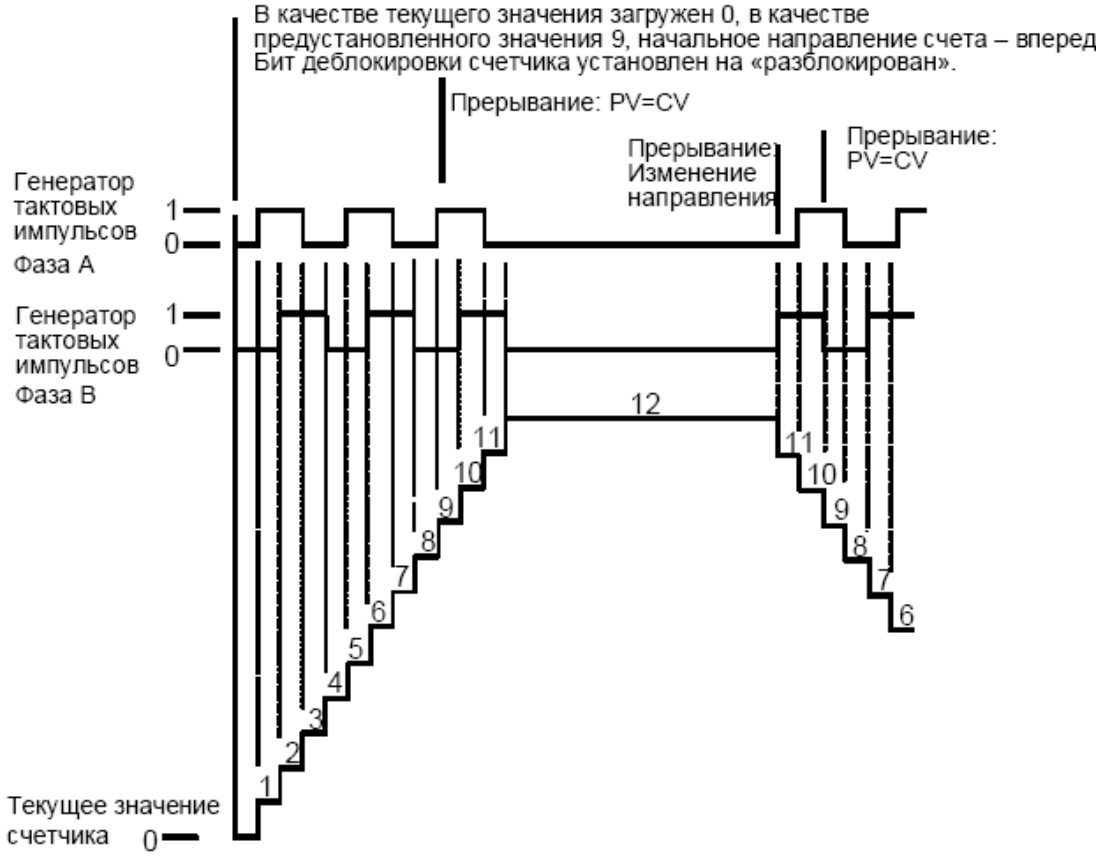


Рис. 21. Пример работы в режимах 9, 10 или 11 (квадратурный режим, четырехкратная скорость).

Четыре счетчика имеют три управляющих бита, которые используются для конфигурирования активного состояния входов сброса и пуска и для выбора односкоростного или четырехскоростного режима счета (только для квадратурных счетчиков). Эти биты находятся в управляющем байте соответствующего счетчика и используются только тогда, когда выполняется команда HDEF. Эти биты определены в табл. 4.

Таблица 4

Активный уровень для управляющих битов сброса, пуска и выбора однократной или четырехкратной скорости

HSC0	HSC1	HSC2	HSC4	Описание (используются только при исполнении HDEF)
SM37.0	SM47.0	SM57.0	SM147.0	Активный уровень управляющего бита для сброса [1]: 0 = сброс активен при высоком уровне; 1 = сброс активен при низком уровне
---	SM47.1	SM57.1	---	Активный уровень управляющего бита для пуска [1]: 0 = пуск активен при высоком уровне; 1 = пуск активен при низком уровне
SM37.2	SM47.2	SM57.2	SM147.2	Выбор скорости счета для квадратурных счетчиков: 0 = 4-кратная скорость 1 = 1-кратная скорость
1 По умолчанию входы сброса и пуска активны при высоком уровне сигнала, а в квадратурных счетчиках скорость счета установлена четырехкратной (по отношению к частоте входного датчика тактовых импульсов).				

Необходимо установить эти управляющие биты в соответствии с желаемым состоянием до исполнения команды HDEF. В противном случае счетчик принимает конфигурацию, определенную по умолчанию для выбранного режима работы счетчика.

Если команда HDEF была выполнена, нельзя изменить настройку счетчика, не переведя сначала S7–200 в состояние STOP.

Определив счетчик и режим его работы, можно программировать динамические параметры счетчика. Каждый скоростной счетчик имеет управляющий байт, который позволяет выполнить следующие действия:

разблокировать или заблокировать счетчик;

управлять направлением (только для режимов 0, 1 и 2) или устанавливать начальное направление счета для всех остальных режимов;

загружать текущее значение;

загружать предустановленное значение.

Проверка управляющего байта и соответствующих текущего и предустановленного значений производится при выполнении команды HSC. В табл.5 описан каждый из этих управляющих битов.

Таблица 5

Управляющие биты для скоростных счетчиков

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Описание
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Бит управления направлением счета: 0 = обратный счет 1 = прямой счет
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Записать направление счета в HSC: 0 = не актуализировать 1 = актуализировать направление
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Записать новое предустановленное значение в HSC: 0 = не актуализировать; 1 = актуализировать предустановленное значение
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Записать новое текущее значение в HSC: 0 = не актуализировать; 1 = актуализировать текущее значение
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Разблокировка HSC: 0 = заблокировать HSC; 1 = разблокировать HSC

Каждый скоростной счетчик имеет 32-битное *текущее значение* и 32-битное *предустановленное значение*. Оба значения являются целыми числами со знаком. Чтобы загрузить новое текущее или предустановленное значение, необходимо настроить управляющий байт и байты специальной памяти, содержащие текущее и/или предустановленное значение, а также выполнить команду HSC, чтобы новые значения были переданы в скоростной счетчик.

Табл.6 описывает байты специальной памяти, используемые для хранения новых текущих и предустановленных значений.

В дополнение к управляющим байтам и байтам, содержащим новые текущие и предустановленные значения, текущее значение каждого скоростного счетчика может быть прочитано путем задания типа данных HC (текущее значение скоростного счетчика), за которым следует номер (0, 1, 2, 3, 4 или 5) счетчика, как показано в табл.7. Текущее значение непосредственно доступно для операций чтения, но оно может быть записано только с помощью команды HSC.

Таблица 6

Новое текущее и новое предустановленное значение

Загружаемое значение	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Новое текущее значение	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
Новое предустановленное значение	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

Таблица 7

Текущие значения скоростных счетчиков

Значение	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Текущее значение	HC0	HC1	HC2	HC3	HC4	HC5

Для доступа к счетному значению скоростного счетчика указывается адрес этого счетчика с помощью типа памяти (HC) и номера счетчика (например, HC0). Текущее значение скоростного счетчика доступно только для чтения и может быть адресовано только как двойное слово (32 бита), как показано на рис. 22.

Рис. 22. Доступ к текущему значению скоростного счетчика.

Все режимы счетчиков поддерживают *прерывание* по равенству текущего значения HSC загруженному предустановленному значению. Режимы счетчиков, использующие вход внешнего сброса, поддерживают прерывание по активизации внешнего сброса. Все режимы счетчиков, кроме режимов 0, 1 и 2, поддерживают прерывание по изменению направления счета. Каждое из этих условий возникновения прерываний может быть заблокировано или разблокировано по отдельности. Полностью использование прерываний обсуждается в разделе о командах обмена данными и прерывания.

Каждому скоростному счетчику поставлен в соответствие *байт состояния*, предоставляющий в распоряжение биты памяти, указывающие текущее направление счета, а также информацию о том, действительно ли текущее значение больше или равно предустановленному. Табл. 8 определяет эти биты состояния для каждого скоростного счетчика.

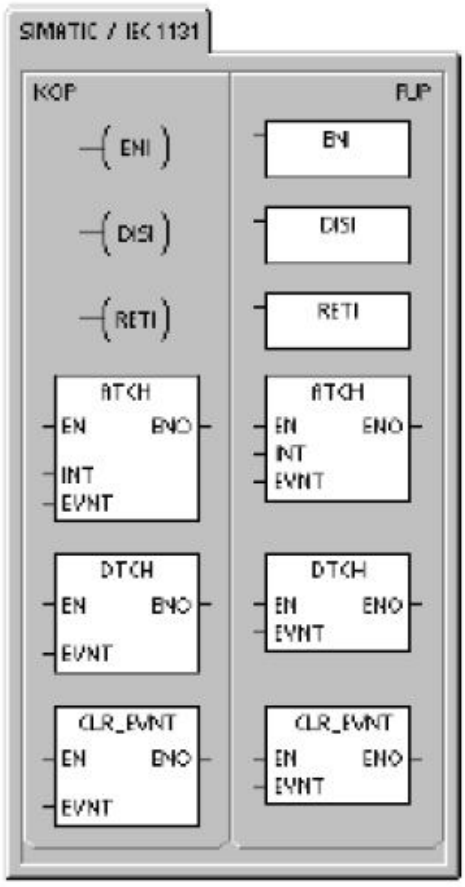
Биты состояния действительны только во время исполнения программы обработки прерывания скоростного счетчика. Цель контроля состояния скоростного счетчика состоит в том, чтобы разблокировать прерывания для событий, оказывающих воздействие на выполняемую операцию.

Таблица 8

Биты состояния для скоростных счетчиков

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Описание
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	Не используются
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	Не используются
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	Не используются
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	Не используются
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	Не используются
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Бит состояния текущего направления счета: 0 = обратный счет 1 = прямой счет
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Бит состояния, указывающий, равно ли текущее значение предустановленному: 0 = не равно 1 = равно
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Бит состояния, указывающий, больше ли текущее значение, чем предустановленное: 0 = меньше или равно 1 = больше

Команды прерывания Siemens S7-200



Разблокирование и блокирование прерываний выполняются командами ENI и DISI. Когда производится перевод в режим RUN, прерывания первоначально заблокированы. Находясь в режиме RUN, можно разблокировать все прерывания, выполнив команду разблокирования прерываний.

Выполнение команды блокирования прерываний запрещает обработку прерываний, однако активные прерывающие события и далее будут ставиться в очередь.

Команда условного возврата из программы обработки прерываний (CRETI) может быть использована для возврата из программы обработки прерываний в зависимости от условия, задаваемого предшествующей логикой.

Команда назначения прерывания (ATCH) связывает прерывающее событие EVNT с номером программы обработки прерываний INT и разблокирует прерывающее событие.

Команда отсоединения прерывания (DTCH) разрывает связь прерывающего события EVNT со всеми программами обработки прерываний и блокирует прерывающее событие.

Команда очистки прерывающих событий удаляет все прерывающие события типа EVNT из очереди прерываний. Эта команда используется для очистки очереди прерываний от нежелательных прерывающих событий. Если эта команда используется для удаления ложных прерывающих событий, вы должны отсоединить это событие перед удалением событий из очереди. Иначе после выполнения команды очистки прерывающих событий к очереди будут добавлены новые события.

Прежде чем программа обработки прерывания может быть вызвана, должно быть установлено соответствие между прерывающим событием и сегментом программы, который нужно выполнить, когда это событие происходит. Для организации связи между прерывающим событием (задаваемым номером прерывающего события) и сегментом программы (задаваемым номером программы обработки прерывания) используется команда назначения прерывания (ATCH). Одной программе обработки прерываний можно поставить в соответствие несколько прерывающих событий, но одно событие не может быть одновременно поставлено в соответствие нескольким программам обработки прерываний.

При назначении прерывающего события программе обработки прерываний, это прерывание автоматически разблокируется.

Отдельные прерывающие события можно заблокировать разрывом связи между этим прерывающим событием и программой обработки прерывания с помощью команды отсоединения прерывания. Команда отсоединения возвращает прерывание в неактивное или игнорируемое состояние. Табл. 9 перечисляет различные типы прерывающих событий.

Программа обработки прерывания исполняется в ответ на соответствующее внутреннее или внешнее событие. После выполнения последней команды программы обработки прерывания управление возвращается в главную программу. Обработка прерываний обеспечивает быструю реакцию на определенные внутренние или внешние события.

В программе обработке прерывания нельзя использовать команды блокирования прерываний (DISI), разблокирования прерываний (ENI), определения режима работы скоростного счетчика (HDEF) и завершения обработки (END).

Так как прерывания могут оказывать влияние на контакты, катушки и аккумуляторы, то система сохраняет и перезагружает логический стек, аккумуляторные регистры и биты специальной памяти (SM), которые отображают состояние аккумуляторов и команд. Это позволяет избежать искажения главной программы пользователя из-за перехода в программу обработки прерывания и возвращения из нее.

Таблица 9

Прерывающие события

Событие	Описание	CPU 221 CPU 222	CPU 224	CPU 224XP CPU 226
0	I0.0 Нарастающий фронт	да	да	да
1	I0.0 Падающий фронт	да	да	да
2	I0.1 Нарастающий фронт	да	да	да
3	I0.1 Падающий фронт	да	да	да
4	I0.2 Нарастающий фронт	да	да	да
5	I0.2 Падающий фронт	да	да	да
6	I0.3 Нарастающий фронт	да	да	да
7	I0.3 Падающий фронт	да	да	да
8	Порт 0 Символ принят	да	да	да
9	Порт 0 Передача завершена	да	да	да
10	Управляемое временем прерывание 0 SMB34	да	да	да
11	Управляемое временем прерывание 1 SMB35	да	да	да
12	HSC0 CV=PV (текущее значение = предустановленному)	да	да	да
13	HSC1 CV=PV (текущее значение = предустановленному)		да	да
14	HSC1 Изменение направления		да	да
15	HSC1 Внешний сброс		да	да
16	HSC2 CV=PV (текущее значение = предустановленному)		да	да
17	HSC2 Изменение направления		да	да
18	HSC2 Внешний сброс		да	да
19	PLS0 Прерывание по завершению отсчета количества импульсов PTO	да	да	да
20	PLS1 Прерывание по завершению отсчета количества импульсов PTO	да	да	да
21	Таймер T32 Прерывание CT=PT	да	да	да
22	Таймер T96 Прерывание CT=PT	да	да	да
23	Порт 0 Прием сообщения завершен	да	да	да
24	Порт 1 Прием сообщения завершен			да
25	Порт 1 Символ принят			да
26	Порт 1 Передача завершена			да
27	HSC0 Изменение направления	да	да	да
28	HSC0 Внешний сброс	да	да	да
29	HSC4 CV=PV (текущее значение = предустановленному)	да	да	да
30	HSC4 Изменение направления	да	да	да
31	HSC4 Внешний сброс	да	да	да
32	HSC3 CV=PV (текущее значение = предустановленному)	да	да	да
33	HSC5 CV=PV (текущее значение = предустановленному)	да	да	да

Из программы обработки прерывания можно вызвать только один уровень вложенности подпрограмм. Аккумуляторы и логический стек совместно используются программой обработки прерывания и вызываемой подпрограммой.

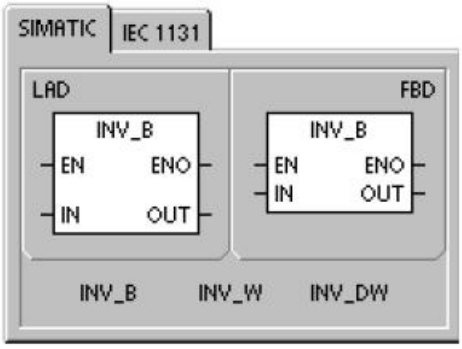
Прерывания обслуживаются S7–200 в порядке их возникновения с учетом соответствующей группы приоритета. В любой момент времени исполняется

только одна программа обработки прерывания. Когда исполнение программы обработки прерывания начинается, программа исполняется до своего завершения. Она не может быть прервана другой программой обработки прерывания, даже если последняя имеет более высокий приоритет. Прерывания, возникающие во время обработки другого прерывания, ставятся в очередь для последующей обработки.

Подробнее о прерываниях смотрите в[2].

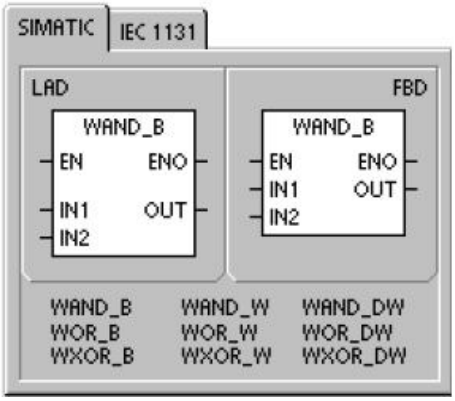
Операции инвертирования

Команды инвертирования байта (INVB), слова (INVW) и двойного слова (INVD) образуют дополнение входа IN до единицы и загружают результат по адресу OUT.



Поразрядные логические операции И, ИЛИ и исключающее ИЛИ

*Поразрядные логические операции И*с байтами (ANDB), словами (ANDW) и двойными словами (ANDD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции И и загружают результат по адресу OUT.

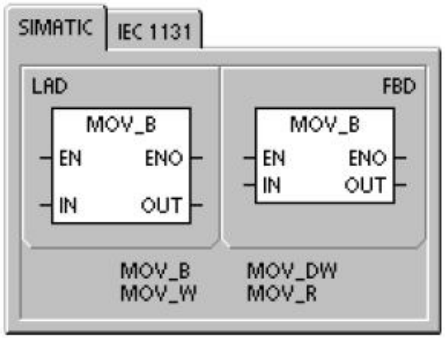


Поразрядные логические операции ИЛИ с байтами (ORB), словами (ORW) и двойными словами (ORD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции ИЛИ и загружают результат по адресу OUT.

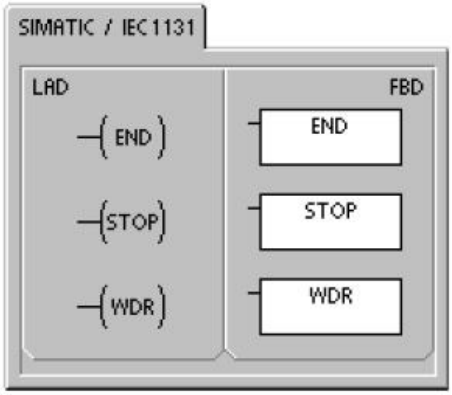
Поразрядные логические операции Исключающее ИЛИ с байтами (XORB), словами (XORW) и двойными словами (XORD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции Исключающее ИЛИ и загружают результат по адресу OUT.

Команды пересылки

Команды пересылки байта (MOVB), слова (MOVW), двойного слова (MOVD) и вещественного числа (MOVR)пересылают значение из адреса IN в адрес OUT, не изменяя исходной величины. Команда пересылки двойного слова используется, в частности, для создания указателя.



Команды управления программой

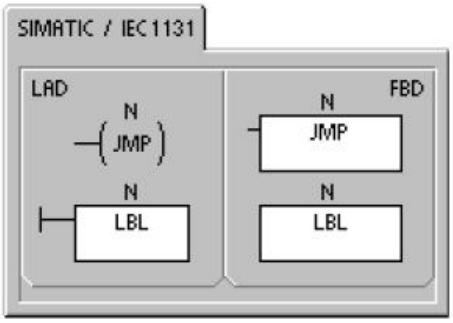


Команда условного завершения (END) завершает текущий цикл в зависимости от результата предшествующей логической операции. Ее можно использовать в главной программе, но не в подпрограммах и программах обработки прерываний.

Команда останова (STOP) завершает выполнение программы, вызывая переход CPU S7–200 из RUN в STOP. Если команда STOP выполняется в программе обработки прерывания, то эта программа завершается немедленно, а все прерывания, стоящие в очереди, игнорируются. Оставшиеся действия в текущем цикле обработки программы завершаются, включая выполнение главной программы пользователя, а переход из RUN в STOP производится в конце текущего цикла.

Команда сброса контроля времени (WDR) перезапускает системный таймер контроля времени CPU S7–200, увеличивая время, которое может занимать цикл обработки программы, не вызывая ошибки контроля времени.

Команды перехода

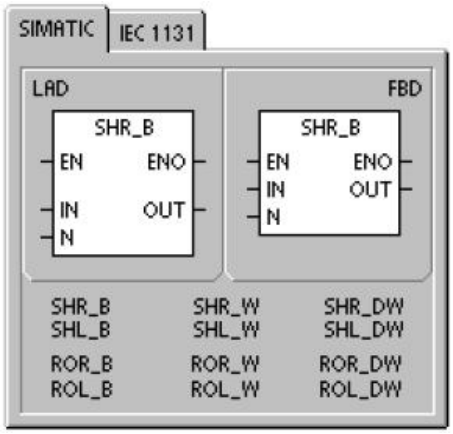


Команда перехода на метку (JMP) осуществляет переход к указанной метке N внутри программы.

Команда «Метка» (LBL) отмечает положение цели перехода N.

Команду перехода на метку можно использовать в основной программе, в подпрограммах и в программах обработки прерываний. Команда перехода и соответствующая метка всегда должны находиться внутри одного и того же сегмента кода (в основной программе, подпрограмме или программе обработки прерываний).

Команды сдвига

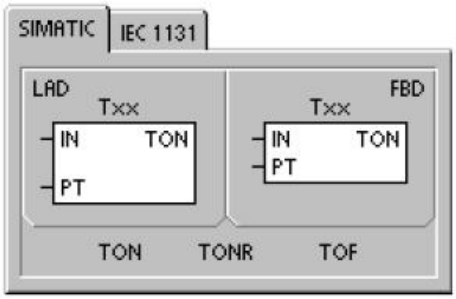


Команды сдвига сдвигают входную величину IN вправо или влево на число разрядов, указанное в N, и загружают результат в выход OUT.

Команды сдвига заполняют позиции выдвигаемых битов нулями. Если величина сдвига (N) больше или равна максимально допустимому значению (8 для операций с байтами, 16 для операций со словами и 32 для операций с двойными словами), то сдвиг производится на максимально возможную величину для данной операции. Если величина сдвига больше 0, то бит переполне-

ния (SM1.1) принимает значение последнего выдвинутого бита. Бит нулевого значения (SM1.0)устанавливается, если результат операции сдвига равен нулю.

Таймерные команды



Команды «Таймер с задержкой включения» (TON) и *«Таймер с задержкой включения с запоминанием»* (TONR) отсчитывают время, когда включен разрешающий вход. Номер таймера (Txx) определяет его разрешающую способность, и эта разрешающая способность отображается в блоке команды.

Таймер с задержкой выключения (TOF) используется для задержки выключения выхода на фиксированный интервал времени после выключения входа.

Как показано в табл. 10, эти три вида таймеров выполняют различные задачи измерения времени:

таймер с задержкой включения TON может использоваться для отсчета отдельного интервала;

таймер с задержкой включения с запоминанием TONR может использоваться для накапливания нескольких отсчитанных интервалов времени;

таймер с задержкой выключения TOF может использоваться для увеличения интервала времени после выключения (или сбоя), например, для охлаждения двигателя после его отключения.

Таблица 10

Работа таймеров

Тип	Текущее время >= предустановленному	Состояние разрешающего входа (IN)	Выключение-включение питания / первый цикл
TON	Бит таймера установлен Отсчет текущего значения продолжается до 32 767	ON: Текущее значение отсчитывает время OFF: Бит таймера сброшен, текущее значение = 0	Бит таймера сброшен Текущее значение = 0
TONR	Бит таймера установлен Отсчет текущего значения продолжается до 32 767	ON: Текущее значение отсчитывает время OFF: Бит таймера и текущее значение сохраняют последнее состояние	Бит таймера сброшен Текущее значение может быть сохранено[1]
TOF	Бит таймера сброшен Текущее время = предустановленному, отсчет времени прекращен	ON: Бит таймера установлен, текущее значение = 0 OFF: Таймер выполняет отсчет времени после перехода из включенного состояния в выключенное	Бит таймера сброшен Текущее значение = 0

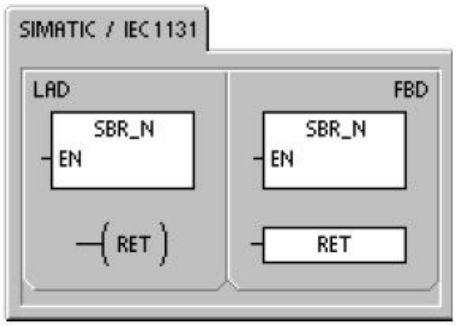
У таймера с разрешающей способностью 1 мс бит таймера и текущее значение обновляются асинхронно с циклом обработки программы. Для циклов, превышающих 1 мс, бит таймера и текущее значение обновляются несколько раз в течение цикла.

У таймера с разрешающей способностью 10 мс бит таймера и текущее значение обновляются в начале каждого цикла обработки программы. Текущее значение и бит таймера остаются неизменными на протяжении цикла, и интер-

валы времени, накопленные в течение цикла, добавляются к текущему значению в начале каждого цикла обработки программы.

У таймера с разрешающей способностью 100 мс бит таймера и текущее значение обновляются, когда выполняется команда; поэтому, чтобы таймер сохранял правильное значение времени, обратите внимание на то, чтобы ваша программа выполняла команду для 100–миллисекундного таймера только один раз за цикл.

Операции с подпрограммами



Команда вызова подпрограммы (CALL) передает управление подпрограмме SBR_N. Команду вызова подпрограммы можно использовать с параметрами или без них. Как только исполнение подпрограммы завершается, управление возвращается команде, следующей за вызовом подпрограммы.

Команда условного возврата из подпрограммы (RET) завершает подпрограмму в зависимости от результата предшествующей логической операции.

Когда вызывается подпрограмма, весь логический стек сохраняется, вершина стека устанавливается в единицу, все остальные ячейки стека устанавливаются в ноль и управление передается вызываемой подпрограмме. Когда эта подпрограмма завершается, стек восстанавливается со значениями, сохраненными в точке вызова, а управление возвращается в вызывающую программу.

Аккумуляторы являются общими для подпрограмм и вызывающей программы. При использовании подпрограммы операции сохранения и восстановления к аккумуляторам не применяются.

Если подпрограмма вызывается в одном и том же цикле несколько раз, то нельзя применять команды «Нарастающий фронт», «Падающий фронт», а также таймеры и счетчики.

Подпрограмма может содержать передаваемые параметры. Параметры определяются в таблице локальных переменных подпрограммы. Параметру должно быть назначено символическое имя (не более 23 символов), тип переменной и тип данных. В подпрограмму и из нее может быть передано шестнадцать параметров.

Поле типа переменной в таблице локальных переменных определяет, передается ли переменная в подпрограмму (IN), в подпрограмму и из нее (IN_OUT), или она передается из подпрограммы (OUT).

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1.

Простейшая программа управления движение лифтовой кабины (2 этажа)

Цель работы

Разработка и реализация программы управления движением лифтовой кабины от первого этажа ко второму и обратно, с остановкой на этажах, открытием и закрытием дверей (с выдержкой времени 3 секунды).

Программа работы

На первом этапе выполнения работы ознакомьтесь с материалом, представленном во *Введении* и создайте программу управления движением без работы дверей.

Предположим, в исходном положении лифтовая кабина находится на первом этаже. Далее главный привод работает в цикле:

1) подъем на максимальной скорости ко второму этажу;

2) при срабатывании межэтажного датчика подхода ко второму этажу в направлении вверх– переход на пониженную скорость;

3) при срабатывании этажного датчика второго этажа – остановка;

4) пауза 3 сек;

5) спуск на максимальной скорости к первому этажу;

6) при срабатывании межэтажного датчика подхода к первому этажу – переход на пониженную скорость;

7) при срабатывании этажного датчика второго этажа– остановка;

8) пауза 3 сек;

9) переход к шагу 1.

Входными сигналами для контроллера будут являться сигналы этажных и межэтажных датчиков (1, 2 этажи).

Выходами контроллера(входами преобразователя частоты) будут*:*

сигнал запуска привода, выход Q0.0;

сигнал реверса, выход Q0.1;

сигнал перехода на пониженную скорость, выход Q0.2.

Логика работы программы:

Контроль движения вверх и вниз (Network 1):

если установлен бит Q0.1 (т.е. осуществляется движение вверх) и не сработал этажный датчик второго этажа,

или,

если не установлен бит Q0.1 (т.е. осуществляется движение вниз) и не сработал этажный датчик первого этажа,

включаем выход Q0.0 (пускаем привод).

Контроль выдержек времени (Network 2,3):

Network 2:

если установлен Q0.1 (т.е. ранее осуществлялось движение вверх) *и*сработал этажный датчик второго этажа

или

не установлен Q0.1 (т.е. ранее осуществлялось движение вниз) *и*сработал этажный датчик первого этажа,

включаем таймер.

Network3:

если сработал таймер:

сбрасываем бит перехода на пониженную скорость Q0.2 (подготавливая привод к движению с максимальной скоростью);

если установлен бит Q0.1, *сбрасываем* его, выходим из текущего цикла программы по инструкции (END) – для того, чтобы не выполнять следующее,

если не установлен бит Q0.1, *устанавливаем* его.

На следующем цикле контроллера главный привод будет включен согласно логике *контроля движения вверх и вниз*.

Контроль скорости (Network 4):

если установлен Q0.1 (движение вверх) *и* сработал межэтажный датчик подхода ко второму этажу в направлении вверх,

или,

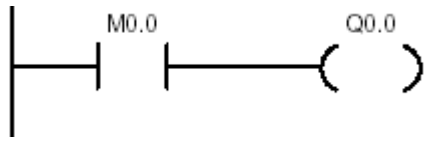
если не установлен Q0.1 (движение вниз) *и* сработал межэтажный датчик подхода к первому этажу,

устанавливаем выход контроллера Q0.2 (тем самым, осуществляя переход на пониженную скорость).

Здесь использована следующая терминология (примеры поясняются гипотетическим кодом на языке С):

1. «Включить» – использовать «обыкновенную» «катушку» –( )– .

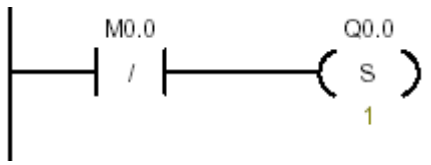
Пример: *если установлен бит M0.0, включить выход Q0.0 (если не установлен – выключить!)*:




```
if (M0.0)
   Q0.0 = 1;
else
   Q0.0 = 0;
```

2. «Установить» – использовать «катушку» с запоминанием состояния «1» –(S)–.

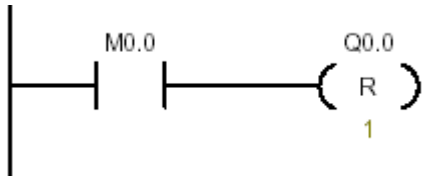
Пример: *если сброшен бит M0.0, установить выход Q0.0 (в противном случае – выход не изменять!)*:




```
if (!M0.0)
   Q0.0 = 1;
```

3. «Сбросить» – использовать «катушку» с запоминанием состояния «0» –(R)–.

Пример: *если установлен бит M0.0, сбросить выход Q0.0 (в противном случае – выход не изменять!)*:




```
if (M0.0)
   Q0.0 = 0;
```

Под катушками –(S)– и –(R)– указывается число бит, начиная с адреса, указанного над катушками, которые нужно установить в единицу или сбросить в нуль соответственно.

Переведите в ручном режиме кабину лифта в промежуточное положение между межэтажными датчиками, запустите программу управления и убедитесь в правильности ее работы. При этом вполне возможно, что система ведет себя непредусмотренным образом не из-за того, что программа управления составлена неверно, а вследствие неправильной настройки параметров преобразователя частоты. В таком случае требуется произвести настройку ПЧ.

Поведение системы зависит в основном от следующих настроек ПЧ: закон частотного управления, вид пуска и торможения, время разгона/торможения.

Закон частотного управления обязательно должен быть установлен линейным (U/f = const), поскольку лифт, безусловно, является подъемным механизмом. IR-компенсация должна быть включена на уровне 50-60В.

Вид разгона– ACCBOOST(автоматическое увеличение пускового момента).

Вид торможения – DECRAMP(постоянное замедление).

Форма функции ускорения/замедления – SLOWS.

Время ускорения – ACCTIME 1 = 2 сек.

*Время замедления*DECTIME 1 = 3 сек.

Первая пониженная скорость CONf1 = 10 Гц.

Значения трех последних параметров обеспечивают «грубую»работу системы (долгое движение на пониженной скорости при подходе к этажу). Подкорректируйте эти значения. При необходимости можно задействовать торможение двигателя главного привода постоянным током.

На втором этапе модернизируйте программу управления установкой, внеся в нее контроль и управление дверями. В целом для этого потребуется:

создать программу Doors, которая будут отвечать за управление дверями;

модифицировать основную программу, предусмотрев, в частности, вместо вызова таймера вызов программы Doors.

Перед программированием рассмотрим необходимость использования дополнительных переменных.

Программы для ПЛК должны уметь распознать, а точнее, запоминать текущее состояние процесса. Так, логика управления приводом при движении кабины вверх отличается от логики управления при движении вниз: в одном случае анализируется сигналы одних датчиков, в другом – других. В первом варианте программы (без управления дверями) дополнительных переменных для запоминания состояния не потребовалась, так как само направления движения «сохраняется» в программе в виде значения выходной переменной Q0.1 (ре-

верс), которое и анализируется во всех четырех фрагментах (*Networks*)программы.

В случае с дверями без дополнительных переменных не обойтись.

Во-первых, при нахождении кабины на этаже при закрытых дверях программа должна знать, отрывались ли на этом этаже двери или нет. Если двери еще не открывались, нужно вызвать программу Doors, в противном случае следует произвести реверс и двигаться к другому этажу. Пусть факт «срабатывания» дверей фиксируется в битовой переменной M0.0 Этот бит будет устанавливаться в программе Doors после всех манипуляций с дверьми и сбрасываться (одновременно с реверсом) в основной программе.

Во-вторых, программе Doors нужно различать фазу открытия дверей от фазы их закрытия. Это позволит правильно организовать управление приводом дверии таймером задержки. Пусть за распознавание фазы работы дверей отвечает бит M0.1: если он не установлен, производится открытие дверей и последующая выдержка времени, если M0.1=1, двери закрываются.

Программа Doors

Network 1:

если не установлен бит M0.1 («первая фаза» работы дверей) *и* не сработал концевой выключатель в положении «Открыто», *включить* открытие дверей.

Network 2:

если не установлен бит M0.1 *и* сработал концевой выключатель в положении «Открыто», *включить* таймер.

Network 3:

если таймер сработал, *установить* бит M0.1 (перейти во «вторую фазу» работы дверей).

Network 4:

если установлен бит M0.1 («вторая фаза» работы дверей) *и* не сработал концевой выключатель в положении «Закрыто», *включить* закрытие дверей.

Network5:

если установлен бит M0.1 *и* сработал концевой выключатель в положении «Закрыто»,

сбросить бит M0.1 (подготавливаясь к следующему циклу);

установить бит M0.0 (сообщая тем самым основной программе, что текущий цикл работы дверей завершен).

Основная программа:

в*Network 2* вместо вызова таймера сделать вызов программы Doors;

в*Network 3* анализировать не состояние таймера, а значение бита M0.0 (выполнять «дальнейшие» действия, если он установлен), кроме того, предусмотреть здесь же *сброс* этого бита (для подготовки будущих циклов).*Сброс бита обязательно должен быть осуществлен до выполнения инструкций управления реверсом, поскольку они содержат инструкцию (END).*

Для обеспечения надежности работы программы добавить *Network*, в котором на первом цикле контроллера (если установлен бит SM0.1) *сбросить*биты M0.0 и M0.1.

Содержание отчета

1. Структурная схема лабораторного стенда.
2. Принципиальные схемы соединений.
3. Параметры настройки преобразователя частоты.
4. Программа управления по варианту 1.
5. Программа управления по варианту 2.

Контрольные вопросы

1. Опишите основные элементы лабораторного стенда.
2. Опишите схему соединений входных цепей контроллера.
3. Опишите схему соединений выходных цепей контроллера.
4. Опишите сигналы управления преобразователем частоты.
5. Какие параметры преобразователя частоты требуют настройки для правильной работы системы управления?
6. Поясните цепь управления включением главного привода.
7. Поясните цепь управления реверсом.
8. Поясните цепи управления скоростью.
9. Какие дополнительные переменные потребовались для организации управления дверями лифтовой кабины?
10. Какие изменения внесены в главную программу во втором варианте?
11. Опишите логику работы подпрограммы Doors.

Лабораторная работа № 2.

Грузовой режим работы лифта (4 этажа)

Цель работы

Разработка и реализация программ управления циклическим движением лифтовой кабины между четырьмя этажами с открытием и закрытием дверей (3 варианта).

Программа работы

Вариант 1. Циклическое движение с остановками на каждом этаже при движении как вверх, так и вниз

В отличие от предыдущей программы (см. лаб. раб. №1) выходной сигнал Q0.1 (реверс) уже не может использоваться в качестве условия для управления включением главного привода и вызова программы Doors, так как реверс должен осуществляться не на каждом этаже, а только на крайних. Далее в общем виде описана логика работы программы.

Главный привод должен включаться, если кабина не находится *ни на одном* из этажей *или* программа Doors отработала (бит M0.0 установлен).

Программа управления дверями должна быть вызвана, если она еще не отработала (бит M0.0 сброшен) *и* кабина находится *на каком-либо* из этажей. По этому же условию можно сбросить бит перехода на пониженную скорость Q0.2.

Бит запоминания отработки дверей M0.0 должен быть сброшен, когда кабина не находится *ни на одном* из этажей.

Бит перехода на пониженную скорость Q0.2 должен быть *установлен*, если кабина движется вверх и сработал *один из* межэтажных датчиков подхода кабины к этажу в направлении вверх или если кабина движется вниз и сработал *один из* межэтажных датчиков подхода кабины к этажу в направлении вниз.

Бит управления направлением Q0.1 должен быть *установлен* после отработки дверей на первом этаже, а *сброшен* на четвертом.

Вариант 2. Циклическое движение с остановками при движении вверх

Данный вариант может быть полезен, например, при развозе грузов с первого этажа (склада) на верхние этажи.

Разработанная в первом варианте программа требует незначительной модификации.

Главный привод должен включаться, если кабина

двигаясь вверх, не находится *ни на* втором, *ни на* третьем, *ни на* четвертом этаже *или,*

двигаясь вниз, не находится на первом этаже *или*

программа Doors отработала (бит M0.0 установлен).

Программа управления дверями должна быть вызвана, если она еще не отработала (бит M0.0 сброшен) *и* кабина,

двигаясь вверх, находится на втором *или* на третьем *или* на четвертом этаже, *или*,

двигаясь вниз, находится на первом этаже.

По этим же условиям можно сбросить бит перехода на пониженную скорость Q0.2.

Бит запоминания отработки дверей M0.0 должен быть сброшен, когда кабина не находится *ни на одном* из этажей.

Бит перехода на пониженную скорость Q0.2 должен быть *установлен*, если кабина движется вверх и сработал *один из* межэтажных датчиков подхода кабины к этажу в направлении вверх *или* если кабина движется вниз и сработал межэтажный датчик подхода кабины к первому этажу.

Бит управления направлением Q0.1 должен быть *установлен* после отработки дверей на первом этаже, а *сброшен* на четвертом.

Вариант 3. Циклическое движение с остановками при движении вниз

Данный вариант может быть полезен, например, при развозе грузов со всех верхних этажей на первый этаж (склад).

Разработайте данную программу самостоятельно, по аналогии с предыдущей.

Содержание отчета

1. Программа управления по варианту 1.
2. Программа управления по варианту 2.
3. Программа управления по варианту 3.

Контрольные вопросы

1. Опишите логику управления включением главного привода в программе управления по варианту 1.
2. Поясните цепи управления реверсом по варианту 1.
3. Поясните цепи управления скоростью по варианту 1.
4. Опишите логику вызова подпрограммы Doors по варианту 1.
5. Опишите логику управления включением главного привода в программе управления по варианту 2.
6. Поясните цепи управления реверсом по варианту 2.
7. Поясните цепи управления скоростью по варианту 2.
8. Опишите логику вызова подпрограммы Doors по варианту 2.
9. Опишите логику управления включением главного привода в программе управления по варианту 3.
10. Поясните цепи управления реверсом по варианту 3.
11. Поясните цепи управления скоростью по варианту 3.
12. Опишите логику вызова подпрограммы Doors по варианту 3.

Лабораторная работа № 3.

Программа управления пассажирским лифтом 4 этажа, простой принцип работы

Цель работы

Разработка и реализация программ управления пассажирским лифтом четырехэтажного здания с раздельной схемой обработки вызовов и приказов, с применением межэтажных датчиков (вариант 1) и счетчика импульсов на валу двигателя (вариант 2).

Программа работы

Раздельная схема обработки вызовов и приказов предполагает, что система реагирует на первый поступивший сигнал и игнорирует во время его выполнения на последующие. Обрабатываемый в данный момент времени вызов или приказ фиксируются загоранием соответствующего светодиода на пульте управления.

Вариант 1. Управление скоростью с применением межэтажных датчиков

Программа будет отрабатывать *цели*. Для фиксирования целей требуется задействовать дополнительную память. Поскольку в данной работе мы намеренно остаемся в рамках битовой логики, выделим следующие биты:

M0.2 – целевой этаж 1;

M0.3 – целевой этаж 2;

M0.4 – целевой этаж 3;

M0.5 – целевой этаж 4.

Эти биты будут устанавливаться путем опроса кнопок пульта и сбрасываться после достижения цели, т.е. после открытия дверей на целевом этаже.

Биты M0.0 и M0.1 будут по-прежнему отвечать за фиксацию факта отработки дверей и фазы работы дверей (открытие/закрытие).

В *Network 1* сбросим 6 бит, начиная с адреса M0.0,при первом запуске программы (когда установлен бит SM0.1).

Далее по нажатию кнопок пульта будем формировать цель, одновременно включая соответствующие светодиоды.

Поскольку реализуется раздельная обработки, цель может быть зафиксирована, только если никаких других целей нет, т.е. не установлен *ни один из битов M0.2 – M0.5*. Это будет общим условием формирования цели.

Если это условие выполняется

и нажата кнопка вызова на первый этаж, устанавливаем выход, управляющий светодиодом подтверждения вызова на первый этаж, фиксируем цель, устанавливая бит M0.2;

и нажата кнопка приказа на первый этаж, устанавливаем выход, управляющий светодиодом подтверждения приказа на первый этаж, и фиксируем цель, устанавливая бит M0.2;и.т.д.

В *Networks 4-6*будем выключать светодиоды после достижения целей. В частности,

если не установлен бит M0.2, сбрасываются выходные биты подтверждения вызова на первый этаж и приказа на первый этаж, и.т.д.

В *Network 7* поместим цепь управления пуском главного привода.

Привод запускается, если двери кабины закрыты *и*

цель – первый этаж*и* кабина не находятся на первом этаже *или*

цель – второй этаж *и* кабина не находятся на втором этаже и.т.д.

В *Network 8* будет вызываться подпрограмма Doors. Условие вызова: главный привод отключен *и* кабина находится *на каком-либо* этаже *и* двери не отработали (бит M0.0 не установлен). Одновременно с вызовом подпрограммы сбросим бит перехода на пониженную скорость.

Далее поместить логику управления битами M0.0 и M0.1. Она будет несколько сложнее, чем в предыдущих работах, поскольку двери кабины пассажирского лифта могут открываться и закрываться на одном и том же этаже более одного раза. Кроме того, в отличие от предыдущих программ предусмотрим защиту от заклинивания дверей.

В *Network 9* сбросим бит M0.0 при наличии *какой-либо* цели. Это позволит вновь запустить цикл открытия/закрытия дверей на следующем целевом этаже или на этом же, если главный привод не был включен в Network 7.

В *Network 10* сбросим бит M0.1 при нахождении кабины на целевом этаже *или* при перегрузке дверей. Благодаря этому станет возможным перейти к открытию дверей, не дожидаясь их полного закрытия в случаях

отказа от поездки (приказом из кабины);

«опоздания» пассажира (вызовом с этажного пульта);

заклинивания дверей.

В *Networks 11,12* сформируем логику перехода на пониженную скорость при движении вверх и вниз соответственно. Она состоит в сопоставлении состояния битов M0.2-M0.5 с сигналами соответствующих межэтажных датчиков.

В *Networks 13,14* поместим цепи установки и сброса сигнала реверса.

Q0.1 *устанавливается,* если

кабина находится на первом этаже *или*

кабина находится на втором этаже *и* целевой этаж третий *или* четвертый *или*

кабина находится на третьем этаже *и* целевой этаж четвертый.

Q0.1 *сбрасывается,* если

кабина находится на четвертом этаже *или*

кабина находится на третьем этаже *и* целевой этаж второй *или* первый *или*

кабина находится на втором этаже *и* целевой этаж первый.

В подпрограмме Doors сделаем единственное дополнение: «одновременно» с запуском таймера *по переднему фронту* сигнала (I2.0 *ине*M0.1) сбросим цели: 4 бита, начиная с адреса M0.2.

При открытии дверей цель считается достигнутой, но во время работы таймера может возникнуть другая цель, поэтому сброс цели должен производиться однократно, по переднему фронту сигнала.

Вариант 2. Управление скоростью с применением счетчика импульсов на валу двигателя

В данном варианте программы переход на пониженную скорость будет осуществляться не по сигналам межэтажных датчиков, а в момент времени, когда скоростной счетчик контроллера отсчитает некоторое число импульсов после достижения этажа, предшествующего целевому. Перед программированием необходимо провести эксперимент для того, чтобы установить это число.

В ручном режиме переведите кабину лифта на первый этаж.

Напишите и запустите следующую программу.

На первом цикле (когда установлен бит SM0.1)запишите управляющий байт счетчика импульсов, поступающих на вход I0.0, выберете режим работы этого счетчика и запустите его.

В *Network 2*обеспечьте включение главного привода в направлении вверх на пониженной скорости, если не сработал этажный датчик второго этажа.

В *Network 3* поместите инструкцию, которая скопирует значение скоростного счетчика в некоторую переменную, например, VD0, в момент срабатывания межэтажного датчика подхода ко второму этажу в направлении вверх. Копирование двойного слова производится инструкцией MOVDW.

Включите режим отображения переменных и после выполнения программы зафиксируйте значение переменной.

Далее определимся с дополнительными переменными, которые потребуются для реализации программы. Пусть в битах M0.6 – M1.1 фиксируется номер этажа, к которому подходит кабина:

M0.6 – кабина подходит к этажу 1;

M0.7 – кабина подходит к этажу 2;

M1.0 – кабина подходит к этажу 3;

M1.1 – кабина подходит к этажу 4.

Это позволит достаточно просто реализовать логику перехода на пониженную скорость.

Внесем в программу, написанную по варианту 1, следующие изменения.

На первом цикле программы сбросим биты M0.0 – M1.1 и подготовим к работе скоростной счетчик: запишем управляющий байт, выберем режим работы и запустим счетчик.

Далее при достижении *какого-либо* из этажей сбрасываем биты M0.6 – M1.1 (подготавливая, таким образом, последующую установку одного из них), обнуляем текущее значение скоростного счетчика и запускаем (перезапускаем) его.

В *Networks 4-6* помещаем логику установки битовM0.6 – M1.1. Так,

если кабина находится на первом этаже, *устанавливаем* бит M0.7 (следующий этаж, безусловно, второй);

если кабина находится на втором этаже *и*

движется (начала двигаться) вверх, *устанавливаем* бит M1.0 (следующий этаж – третий) *или*

движется (начала двигаться) вниз, *устанавливаем* бит M0.6 (следующий этаж – первый)и т.д.

Включение пониженной скорости теперь будет производиться в одном Network (независимо от направления движения). Условие включения пониженной скорости: скоростной счетчик отсчитал заданное количество импульсов *и* кабина подходит к целевому этажу (целевой этаж по прежнему зафиксирован в одном из битов M0.2 – M0.5).

Содержание отчета

1. Программа управления по варианту 1.
2. Программа управления по варианту 2.

Контрольные вопросы

1. Каким образом в программах управления фиксируется целевой этаж?
2. Поясните цепи управления включением светодиодов пульта и фиксации и сброса цели.
3. Поясните цепь управления включением главного привода.
4. Поясните цепь вызова подпрограммы Doors.
5. Опишите логику работы подпрограммы Doors.
6. Поясните цепи управления реверсом.
7. Поясните цепи управления скоростью в программе по варианту 1.
8. Поясните цепь инициализации скоростного счетчика в программе по варианту 2.
9. Как в программе по варианту 2 распознается этаж, к которому подходит кабина?
10. Поясните цепи управления текущим значением скоростного счетчика в программе по варианту 2.
11. Поясните цепи управления скоростью в программе по варианту 2.

Лабораторная работа № 4.

Программа управления пассажирским лифтом многоэтажного здания, собирательный и раздельный принципы работы

Цель работы

Разработка и реализация универсальных программ управления пассажирским лифтом многоэтажного здания с собирательной и раздельной схемой обработки вызовов и приказов, с применением межэтажных датчиков (варианты 1,3) и счетчика импульсов на валу двигателя (варианты 2,4).

Программа работы

В Приложении 1 приведены алгоритмы и программа(на языке С) управления пассажирским лифтом многоэтажного здания, реализующие собирательный принцип работы. В отличие от программ, разработанных ранее, данная программа, во-первых, построена большей частью на обработке *слов*, а не отдельных битов, что делает ее универсальной в плане использования для зданий разной этажности. Во-вторых, программа демонстрирует «автоматный» подход к построению системы управления: она содержит конечный автомат, пребывающий в различных состояниях, формирующий выходные сигналы в зависимости от входных сигналов и текущего состояния и изменяющий свое состояние под воздействием входных сигналов.

Требуется:

1)изучить материал, приведенный в приложении, разработать, отладить и протестировать программу управления для контроллера на языке LAD, реализующую собирательный принцип работы лифта с использованием межэтажных датчиков для управления скоростью главного привода (вариант 1);

2)модифицировать программу управления, заменив межэтажные датчики счетчиком импульсов на валу двигателя (вариант 2);

3) упростить программу управления по варианту 1, перейдя от собирательной к раздельной схеме обработки вызовов и приказов (вариант 3);

4) модифицировать программу управления по варианту 3,заменив межэтажные датчики счетчиком импульсов на валу двигателя (вариант 4).

Содержание отчета

1. Структура программ управления.
2. Блок-схема конечного автомата.
3. Алгоритмы главной программы и подпрограмм по варианту 1.
4. Программа управления по варианту 1.
5. Алгоритмы главной программы и подпрограмм по варианту 2 (только модифицированные части).
6. Программа управления по варианту 2.
7. Алгоритмы главной программы и подпрограмм по варианту 3 (только модифицированные части).

8. Программа управления по варианту 3.
9. Алгоритмы главной программы и подпрограмм по варианту 4 (только модифицированные части).
10. Программа управления по варианту 4.

Контрольные вопросы

1. Каким образом в программах управления кодируется информация о срабатывании датчиков, текущих вызовах и приказах?
2. Опишите состояния, в которых может находиться конечный автомат и логику перехода из одного состояния в другое.
3. Поясните цепи подпрограммы движения к ближайшему этажу.
4. Поясните цепи подпрограммы выбора цели (вариант 1).
5. Поясните цепи подпрограммы движения вверх (вариант 1).
6. Поясните цепи подпрограммы движения вниз (вариант 1).
7. Поясните цепи подпрограммы открытия дверей (вариант 1).
8. Поясните цепи подпрограммы закрытия дверей (вариант 1).
9. Укажите изменения и дополнения в алгоритмах основной программы и подпрограмм, сделанные при переходе к варианту 2.
10. Укажите изменения и дополнения в основной программе и подпрограммах, сделанные при переходе к варианту 2.
11. Укажите изменения и дополнения в алгоритмах основной программы и подпрограмм, сделанные при переходе к варианту 3.
12. Укажите изменения и дополнения в основной программе и подпрограммах, сделанные при переходе к варианту 3.
13. Укажите изменения и дополнения в алгоритмах основной программы и подпрограмм, сделанные при переходе к варианту 4.
14. Укажите изменения и дополнения в основной программе и подпрограммах, сделанные при переходе к варианту 4.

Лабораторная работа № 5.

Имитационная модель системы управления лифтом и визуализация процесса

Цель работы

Разработка имитационной модели для отладки сложных алгоритмов и программ управления лифтовыми механизмами и системы визуализации процесса.

Программа работы

В Приложении 2 помещено описание компьютерной имитационной модели системы управления лифтовыми механизмами, включающая Simulink-модель объекта управления, «виртуальный» контроллер на базе персонального компьютера и программу визуализации и оперативного управления.

На первом этапе требуется изучить материал, приведенный в Приложении и воссоздать имитационную модель. Для этого понадобятся следующие программные пакеты:

MathWorks® MATLAB® (использовалась версия R2011a);

3S-Smart Software® CODESYS® (использовалась версияV2.3);

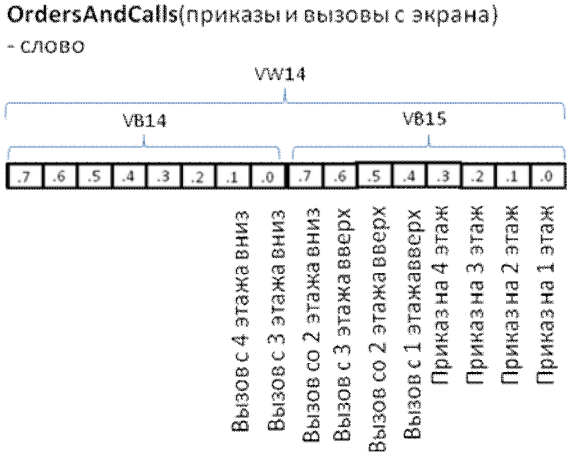
AdAstra Research Group® TRACE MODE® (использовалась версия 6.06).

Далее должна быть построена реальная система управления лабораторной установкой с собирательной схемой обработки вызовов и приказов, применением межэтажных датчиков и компьютерной подсистемой визуализации и оперативного управления на основе экрана визуализации, созданного ранее. Для этого требуется внести необходимые изменения в программу для контроллера, разработанную при выполнении предыдущей лабораторной работы (вариант 1), и проект Trace Mode.

В программе контроллера необходимо сделать следующие изменения.

1) на первом цикле (когда установлен бит SM0.1) инициализировать и запустить скоростной счетчик;

2) в цепях формирования байтовых переменных Orders, Calls_Up, Calls_Down предусмотреть «альтернативные пути» установки битов по сигналам, поступающим от системы визуализации при нажатии экранных кнопок. Эти сигналы можно «сосредоточить» в одной двухбайтовой переменной (слове), например, как показано справа;



3) ввести цепи принудительной установки значений скоростного счетчика при нахождении кабины на этажах. Для определения этих значений потребуется провести эксперимент с применением простейшей тестовой программы, подобной той, что была разработана в лабораторной работе №3.

При достижении кабиной этажа по фронту сигнала (однократно) в переменную счетчика должно быть записано требуемое значение, после чего счетчик должен быть перезапущен;

4) в отдельной цепи поместить набор инструкций, обеспечивающий пересчет значения счетчика в число в диапазоне от 10 до 110 (как в имитационной модели). Для этого нужно:

умножить значение счетчика на 100 и поместить результат в некоторую четырехбайтовую переменную (далее VD8);

поделить VD8 на максимальное значение счетчика (которое он выдает при нахождении кабины на четвертом этаже) и результат вновь поместить в VD8;

добавить к VD8 число 10;

5) обеспечить изменение направления счета при реверсе главного привода. При изменении (по фронту) сигнала «реверс» инвертировать (установить или сбросить) бит управления направлением счета и перезапустить счетчик;

6) сформировать некоторую переменную (слово), например, VW12, для «информирования» системы визуализации о состоянии дверей. В имитационной модели подобная переменная изменяет свое значение от 0 (двери закрыты) до 5 (двери закрыты). На лабораторной установке не предусмотрено непрерывное измерение положения дверей, имеются лишь датчики их состояния, формирующие сигналы «закрыто», «открыто». При полностью закрытых дверях в переменную нужно записать 0, при полностью открытых – 5, в промежуточном состоянии (когда не сработал ни один из датчиков) – некоторое промежуточное значение, например, 3.

В итоге, если все новые переменные были размещены по приведенным выше адресам, получим в переменных:

VD8 – текущее положение кабины в шахте в диапазоне от 10 до 110. При этом фактически заняты только младшие два байта из четырех (старшие содержат нули), поэтому системе визуализации считывать нужно двухбайтовое слово VW10;

VW12 – состояние дверей лифта (возможные значения 0,3 и 5);

VW14 – вызовы и приказы, сформированные с помощью экранных кнопок системы визуализации.

Изменения, которые нужно ввести в проект TraceMode, незначительны:

1)в источниках/приемниках вместо обмена по протоколу OPC необходимо организовать каналы обмена с контроллером Siemens S7 200.Для этого требуется создать группу PLC, далее группу Siemens_PPI_Group и, наконец, компоненты Siemens_PPI. Обмен будет осуществляться посредством следующих переменных:

Calls_Up, Calls_Down, Orders(чтение) – для управления экранными индикаторами вызовов и приказов. Адреса этих переменных см. в программе для контроллера.

VW10 (чтение) – для управления положением экранной «кабины»;

VW12 (чтение) – для управления экранными «дверями»;

VW14 (запись) – для формирования вызовов и приказов с помощью экранных кнопок.

Все компоненты работают с областью Variable Memory(WORD).

Настройка компонента показана на рис. 23

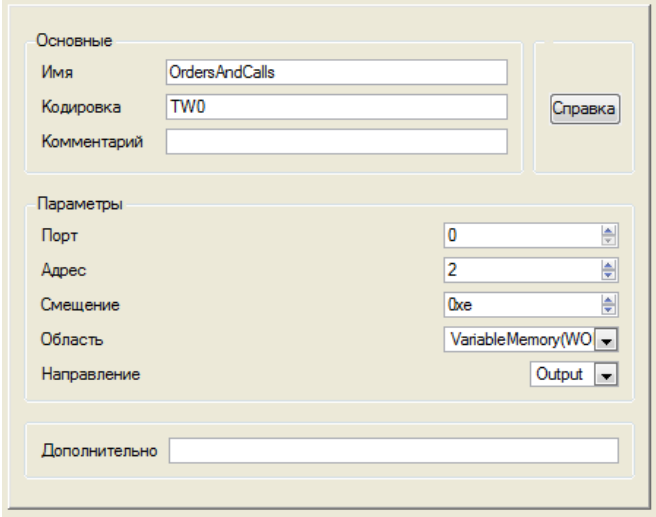


*Рис. 23.*Настройка компонента OrdersAndCalls (VW14, смещение 0xe).

2) создать аргументы экрана и связать их с источниками и приемниками;

3) привязать графические объекты (кабина и ее двери, кнопки и индикаторы) к аргументам экрана.

После сохранения проекта для MPB и создания папки проекта в нее (точнее в папку RTM_1) обязательно нужно поместить файл конфигурации обмена по протоколу PPI `PPI.cfg`. Файл содержит настройки параметров COM-порта, через который компьютер подключен к сети PPI и создается при помощи специальной утилиты `PPIconfig.exe`, которую можно найти в папке

`C:\Program Files\AdAstra Research Group\Trace Mode IDE 6 Base\Drivers_with_Setup\Siemens\PPI.`

После помещения файла проект нужно вновь сохранить для MPB.

Содержание отчета

1. Simulink-модель лифтовых механизмов (с подсистемами).
2. Управляющая программа для PLC WinNT.
3. Экран визуализации Trace Mode.
4. Главная программа для S7-200 с выделением внесенных дополнений.

Контрольные вопросы

1. Какие задачи могут быть решены с помощью имитационной системы?
2. Состав и назначение программных средств имитационной системы.
3. Опишите схему взаимодействия программных компонентов.
4. Какова последовательность действий по запуску OPC-сервера?
5. Опишите Simulink-модель лифтовых механизмов.
6. Опишите подсистему Simulink-модели, моделирующую главный привод лифта.

7. Опишите подсистему Simulink-модели, моделирующую подсистему датчиков.
8. Опишите подсистему Simulink-модели, моделирующую механизм открытия дверей.
9. Укажите отличия программы управления для PLC WinNT от программы контроллера, разработанной в предыдущей работе (вариант 1).
10. Опишите каналы источники/приемники, используемые в системе визуализации, и их настройки.
11. Опишите экран визуализации и его компоненты.
12. Каким образом производится управление движением экранной кабины и открытием/закрытием экранных дверей?
13. Опишите схему взаимодействия экранных индикаторов и кнопок с программой для PLC WinNT.
14. Какие дополнения были внесены в программу контроллера для взаимодействия с системой визуализации?
15. Поясните цепи инициализации и управления скоростным счетчиком.
16. Поясните изменения, сделанные в цепях формирования вызовов и приказов.
17. Поясните цепь пересчета значения скоростного счетчика.
18. Поясните цепи формирования сигнала о состоянии дверей.

ПРИЛОЖЕНИЕ 1.

Алгоритмы и прототип программы управления лифтом: собирательный принцип работы

Описание принципа

Сигналы, которые задают программу движения лифтом, могут поступать как из кабины, так и с лестничных площадок. Схема их обработки может быть раздельной или же собирательной. В первом случае система реагирует на первый поступивший сигнал, и игнорирует во время его выполнения на все последующие. Собирательный же принцип предполагает восприятие уже нескольких команд и их выполнение в оптимальной последовательности.

Собирательный принцип получил широчайшее распространение. На вызывном аппарате лифтов с такой системой управления располагаются две кнопки – вверх и вниз, и при вызове системе управления задается не только этаж, на котором находится вызывающий, но и требуемое направление движения.

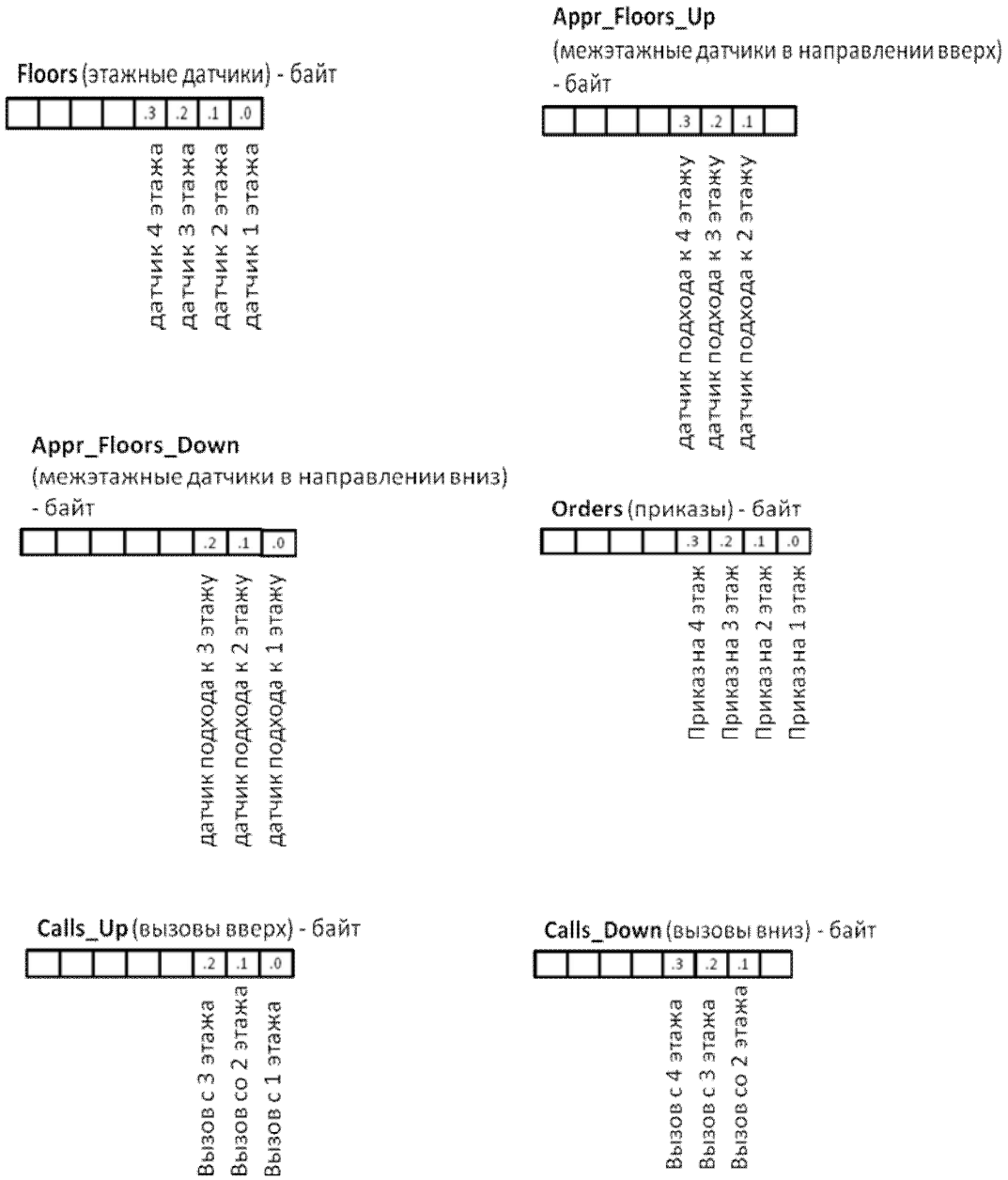
Лифт, движущийся вниз, делает остановки только в соответствии с вызовами вниз, собирая таким образом пассажиров. Вызовы вверх при этом игнорируются (хотя и запоминаются системой управления). При движении вверх игнорируются вызовы вниз.

Решение о реверсе (изменении направления) принимается всегда в «крайних точках»: при движении вверх на самом высоком этаже из всех запрошенных целей движения, по вызову вниз или по приказу; при движении вниз на самом низком этаже из всех запрошенных целей движения, по вызову вверх или по приказу.

Собирательный принцип упорядочивает движение лифта, сокращая до минимума количество челночных движений кабины и экономя тем самым электрическую энергию.

Кодирование

Для того, чтобы сделать программу управления универсальной, т.е. практически независимой от этажности здания, и использовать в ней «мощные» команды, оперирующие байтами и словами, введем следующие переменные:



Эти переменные будут «заполняться» в начале каждого цикла контроллера после опроса входов, связанных с этажными и межэтажными датчиками и кнопками пульта управления.

В нашей реализации используются байты, поэтому максимальная этажность здания, обслуживаемого системой равна восьми. Однако программу можно будет достаточно просто передать и для большего числа этажей, если вместо байтов использовать слова (два байта, 16 этажей) или двойные слова (4 байта, 32 этажа).

Концепция программы

Концепция программы представлена на рис.24.

Главная программа Main:

1) производит опрос этажных и межэтажных датчиков и *копирует* биты соответствующих входов в байтовые переменные Floors, Appr_Floors_Up, Appr_Floors_Down;

2) производит опрос входов, связанных с кнопками пульта и *запоминает* вызовы и приказы, *устанавливая* биты байтовых переменных Calls_Up, Calls_Down, Orders. Сбрасываться эти биты будут только после выполнения соответствующих вызовов и приказов;

3) управляет светодиодами пульта простым *копированием* битов из байтовых переменных Calls_Up, Calls_Down, Ordersв соответствующие выходы контроллера;

4) организует работу конечного автомата, вызывая подпрограммы в зависимости от его состояния, см. рис. 25.

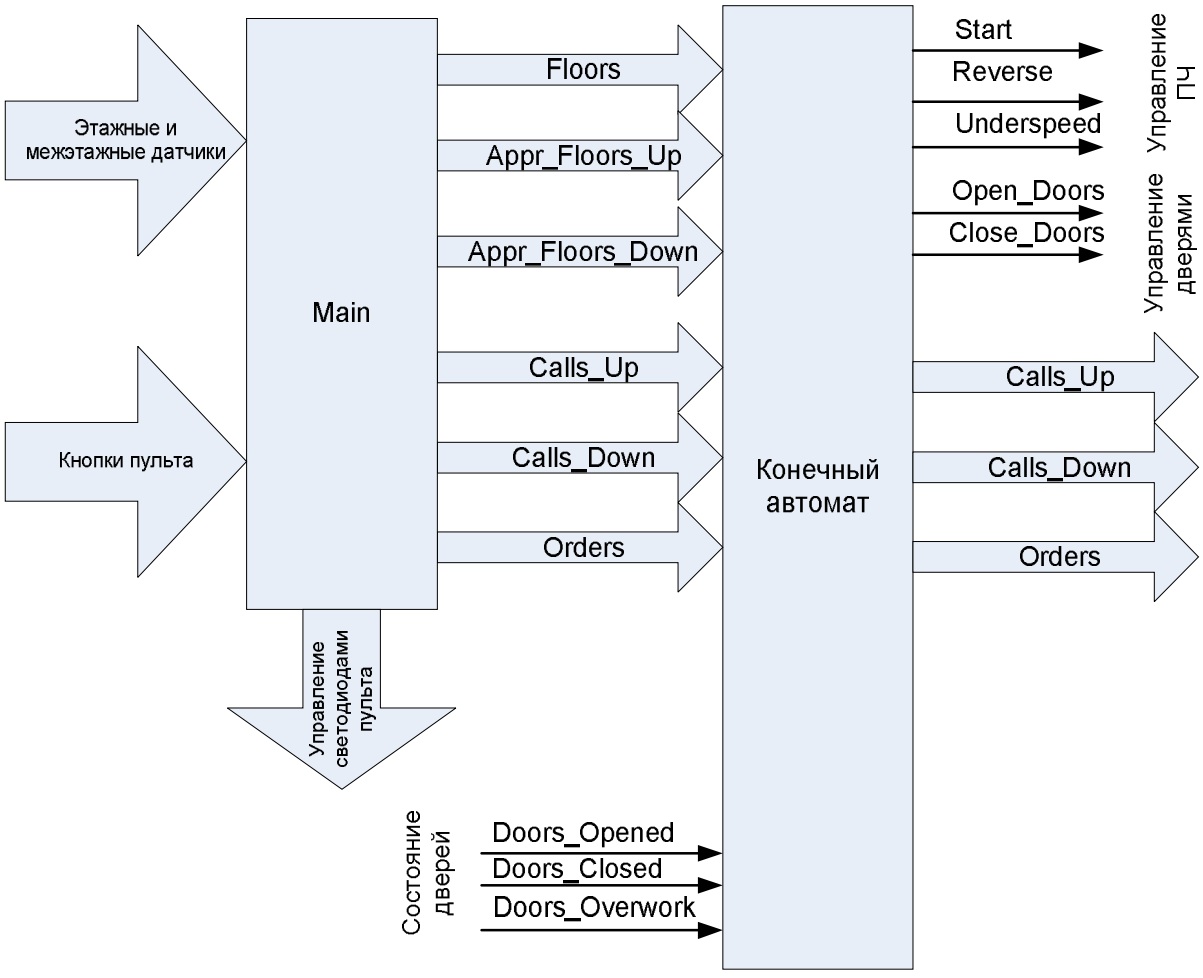


*Рис. 24.*Структура программы.

Конечный автомат формирует команды управления преобразователем частоты (пуск, реверс, переход на пониженную скорость) и приводом дверей кабины (открыть, закрыть) в зависимости от:

1) своего состояния;

2) сигналов этажных и межэтажных датчиков, сгруппированных в переменных Floors, Appr_Floors_Up, Appr_Floors_Down;

3) действующих вызовов и приказов, сгруппированных в переменных Calls_Up, Calls_Down, Orders;

4) сигналов с концевых выключателей дверей Doors_Opened, Doors_Closed и датчика их перегрузки Doors_Overwork.

Помимо этого, конечный автомат сбрасывает биты выполненных вызовов и приказов в байтовых переменных Calls_Up, Calls_Down, Orders.

Структура конечного автомата показана на рис.25.

Главная программа вызывает подпрограммы в зависимости от состояния конечного автомата, которое задается обычным числом от 1 до 6 и фиксируется в переменной state. В начальном состоянии (state=1), которое наступает при запуске системы, вызывается подпрограмма Nearast_Lower_Floor, которая на пониженной скорости доставляет кабину лифта на ближайший этаж в направлении вниз.

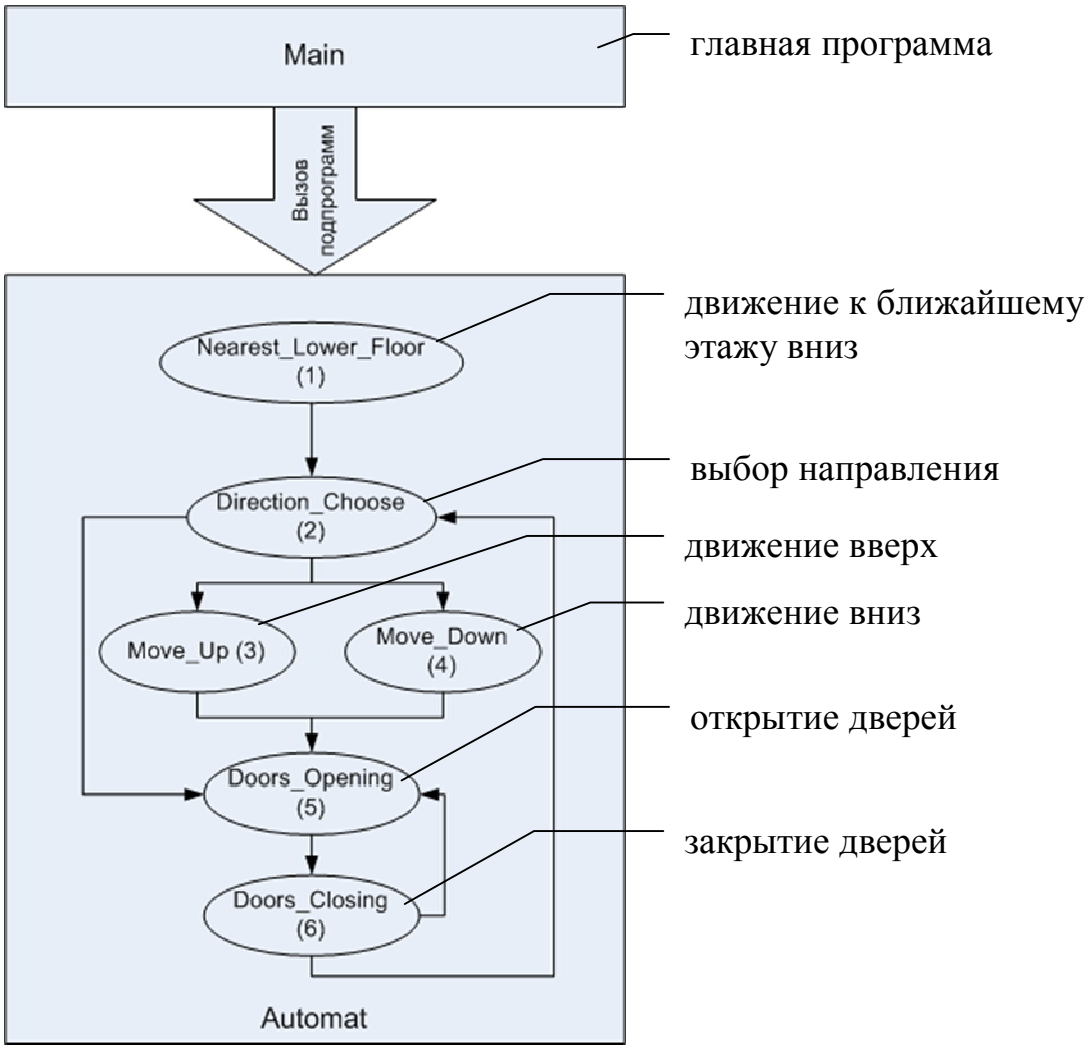


Рис.25. Структура конечного автомата.

Переход из одного состояния в другое подготавливается в самих подпрограммах путем изменения переменной state.

В подпрограмме Direction_Choose (state=2) производится выбор направления движения кабины (вниз или вверх). Переход в данное состояние происходит после закрытия дверей кабины.

ПодпрограммыMove_Up(state=3) иMove_Down(state=4) управляют главным приводом при движении кабины вверх и вниз соответственно.

Подпрограммы Doors_Opening (state=5) и Doors_Closing(state=6) управляют приводом дверей кабины на этажах. Переход из состояния Doors_Closingв состояние Doors_Opening происходит при заклинивании дверей (поступлении сигнала Doors_Overwork).

Подпрограммы

Подпрограммы представлены блок-схемами алгоритмов и «псевдокодом» на языке C/C++.

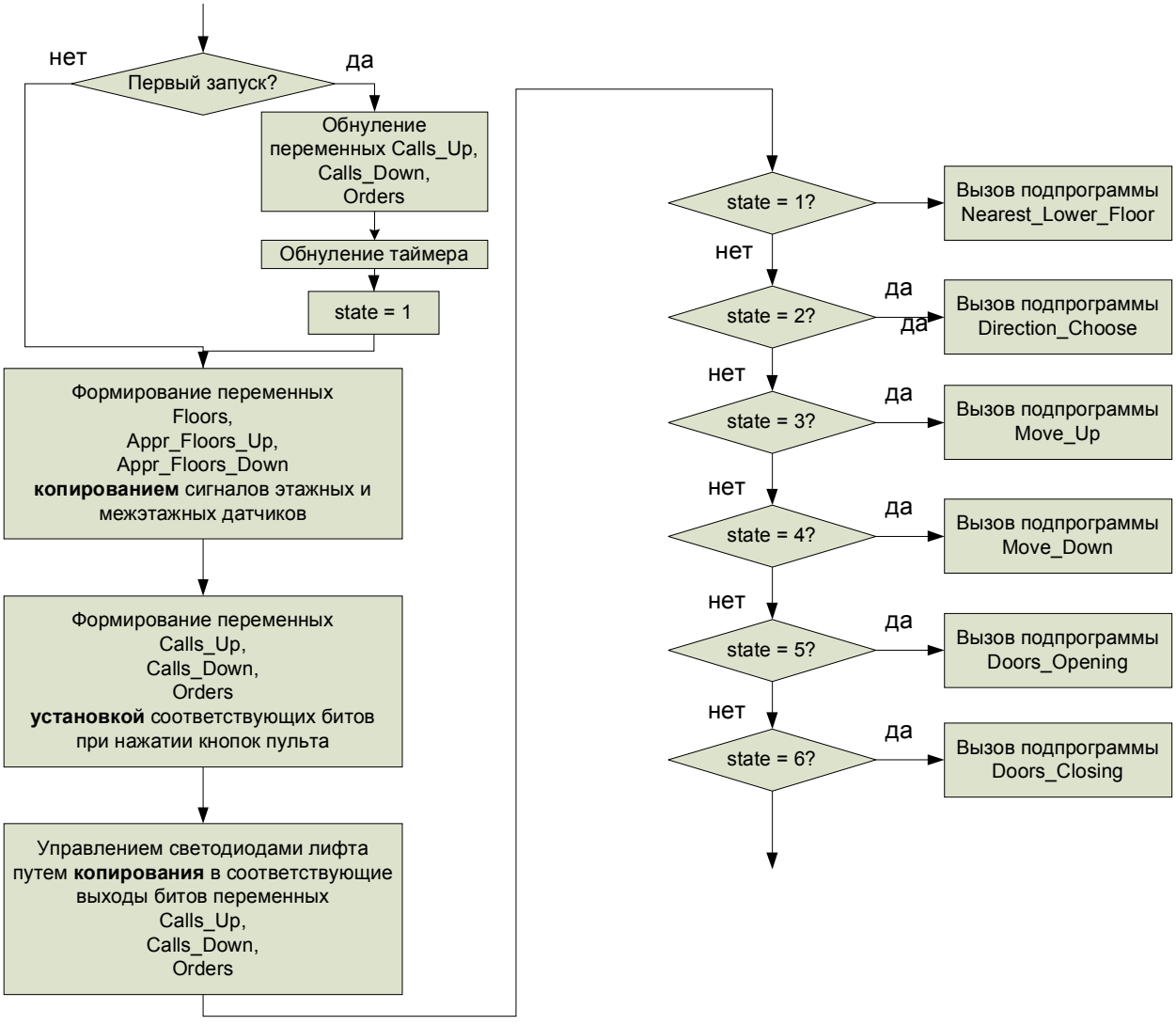


Рис.26. Структура главной программы Main

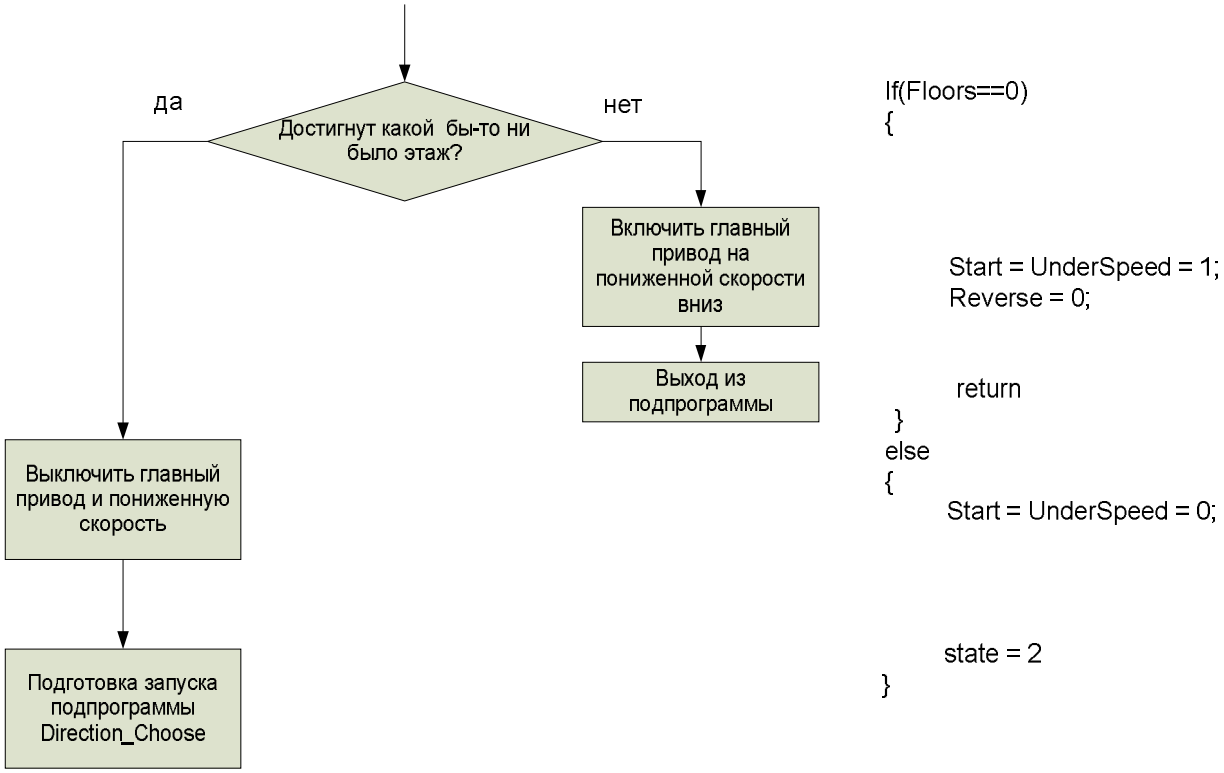
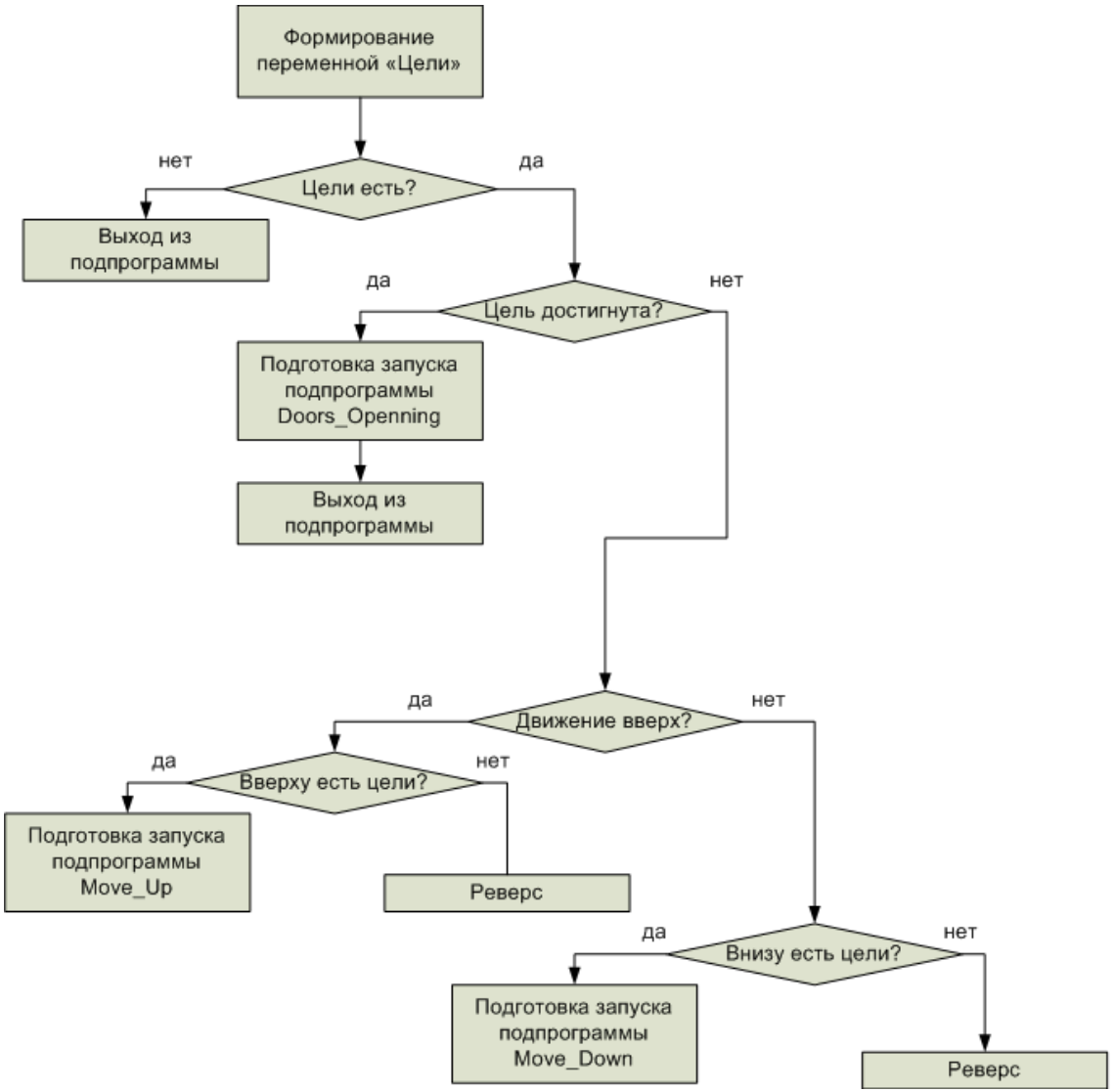
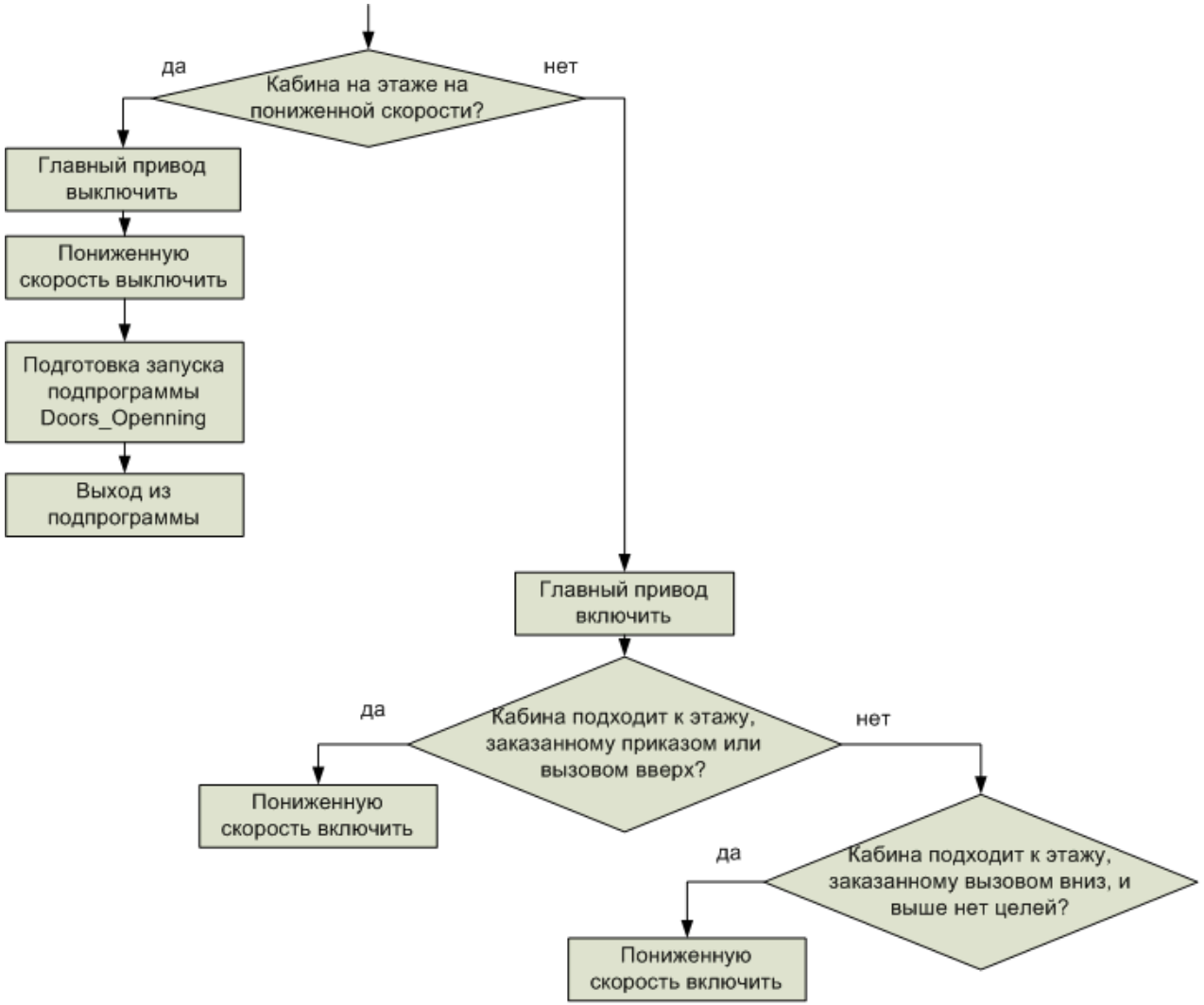


Рис.27. Структура программы Nearast_Lower_Floor



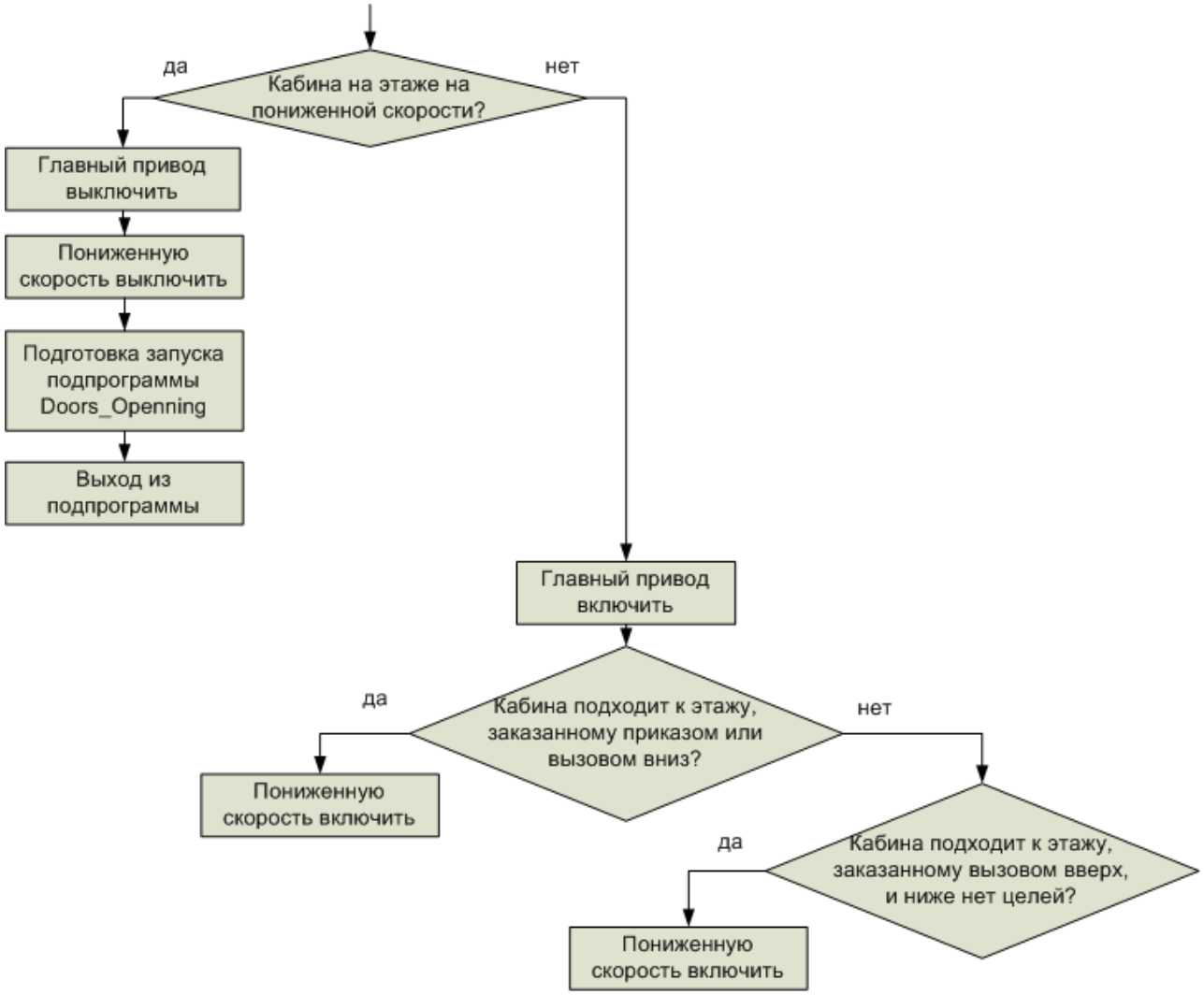

```
unsigned Goals = Calls_Up | Calls_Down | Orders;


If(!Goals)
    return;

If(Goals&Floors)
{
  state = 5;


  return;
}




If(Reverse)

    If(Goals>Floors)
      state = 3;
    else
      Reverse = 0;

else if((Floor-1)&Goals)

      state = 4;
    else
      Reverse = 1;
```

Рис.28. Структура программы Direction_Choose



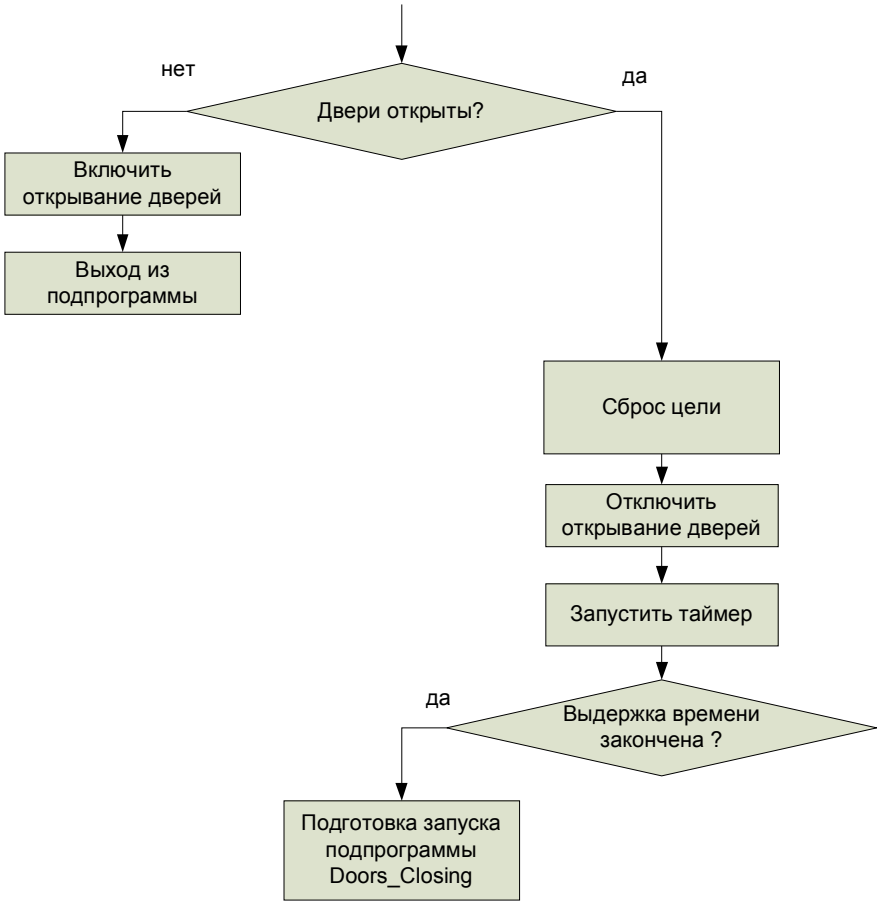

```
unsigned Goals = Calls_Up | Calls_Down | Orders;

If(Floors & UnderSpeed)
{
    Start = 0;


    UnderSpeed = 0;


    state = 5;



    return;
}
else
{
    Start = 1;


    if(Appr_Floors_Up & Orders ||
                Appr_Floors_Up & Calls_Up)

        UnderSpeed = 1;
    else if( (Appr_Floors_Up & Calls_Down) &&
            (Appr_Floors_Up & Calls_Down >
                Appr_Floors^Goals)
            )

        UnderSpeed = 1;
}
```

Рис.29. Структура программы Move_Up.



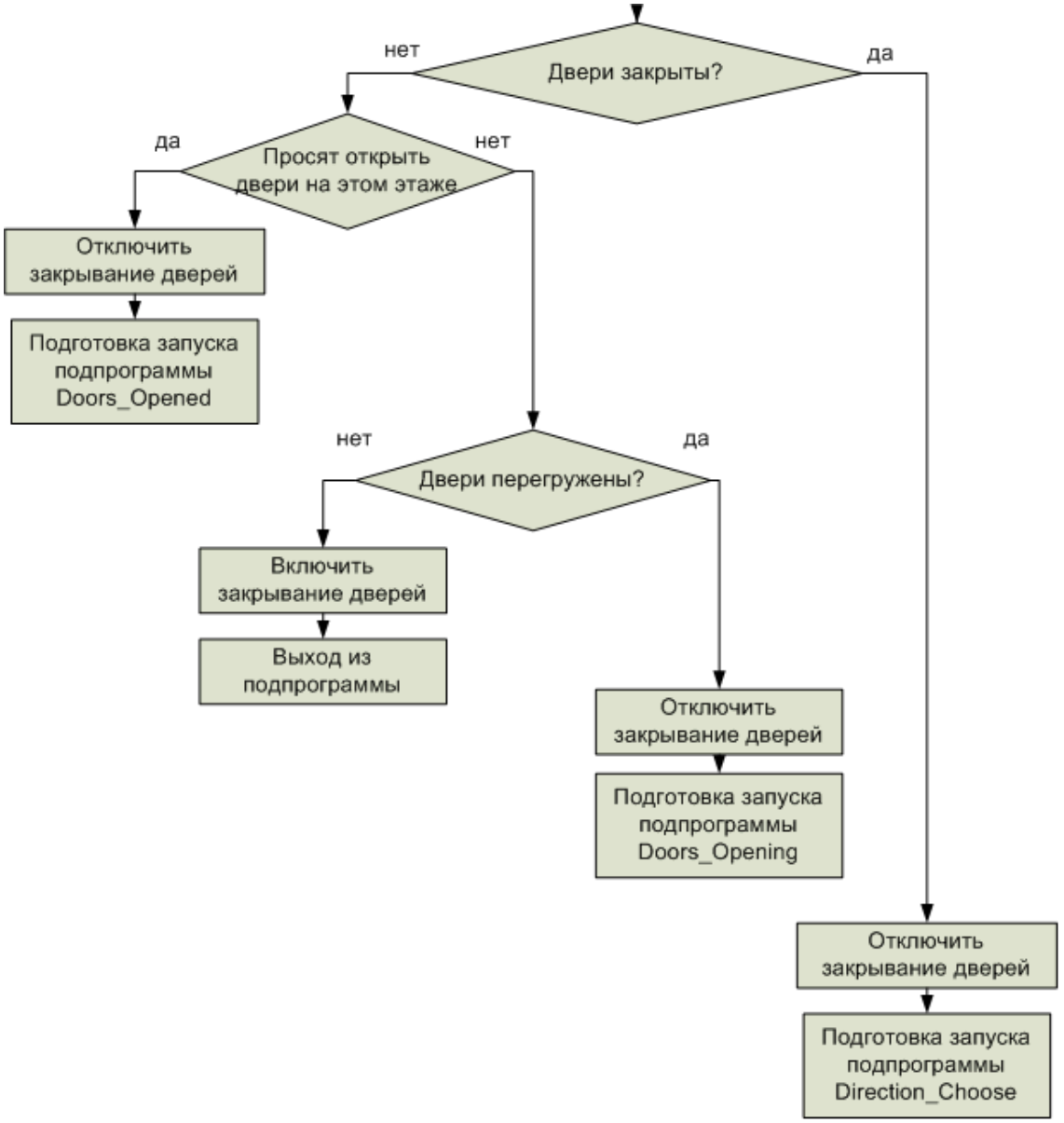

```
unsigned Goals = Calls_Up | Calls_Down | Orders;

  If(Floors & UnderSpeed)
  {

      Start = 0;


      UnderSpeed = 0;


       state = 5;


      return;
  }
  else
  {
      Start = 1;


      if(Appr_Floors_Down & Orders ||
            Appr_Floors_Down & Calls_Down)

          UnderSpeed = 1;
      else if( (Appr_Floors_Down & Calls_Up) &&
              !((Appr_Floors_Down-1) &  Goals)
             )

          UnderSpeed = 1;
  }
```

Рис. 30. Структура программы Move_Down.




```
if (!Doors_Opened)
{
    Open_Doors = 1;

    return;
}
else
{
    Calls_Up&=~Floors;
    Calls_Down&=~Floors;
    Orders&=~Floors;

     Open_Doors = 0;

    Timer(выдержка);
}

if (Timer)

     state = 6;
```

Рис.31. Структура программы Doors_Opening




```
if (!Doors_Closed)
{
    if ((Orders|Calls_Up|Calls_Down)&Floors)
    {

    Doors_Close = 0;

     state = 5;
     return;
     }

    if (!Doors_Overwork)
    {

            Doors_Close = 1;

            return;
    }
    else
    {
           Doors_Close = 0;


           state = 5;
     }
}
else
{
    Doors_Close = 0;


     state = 2;
}
```

Рис.32. Структура программы Doors_Closing.

ПРИЛОЖЕНИЕ 2.

Имитационное моделирование системы управления лифтом

Назначение системы

Разработка АСУ современных технологических процессов – сложная и ответственная задача, решение которой производится в несколько этапов: от составления математической модели до проектирования человеко-машинного интерфейса. Ошибки проектирования АСУ ТП часто трудно исправить на этапе эксплуатации системы, – для этого может потребоваться даже пересмотр базовых концепций, лежащих в ее основе. С другой стороны, ошибки оперативного персонала АСУ могут привести к серьезным последствиям: остановке технологического процесса и авариям оборудования. В связи с этим как проектировщикам, так и оперативному персоналу нужен программный инструмент-симулятор АСУ ТП. Проектировщик с его помощью может решать следующие задачи:

1) имитационное моделирование технологического процесса в различных режимах работы при воздействиях, программно формируемых управляющей аппаратурой и средствами человеко-машинного интерфейса;

2) отладка технологических программ;

3) выбор наиболее удобных для пользователя средств визуализации технологического процесса и способов формирования управляющих воздействий.

Оперативный персонал задействует программный комплекс на этапе настройки АСУ ТП, а также в целях обучения.

В рамках единого комплекса задействуются программные средства разных производителей и классов:

система имитационного моделирования (для построения модели механизма) MathWorks® MATLAB®, Simulink®;

система класса PC-based controller (для программной реализации алгоритмов управления на языках программирования промышленных контроллеров) 3S-Smart Software® CODESYS®, включая PC-эмулятор ПЛК SP PLC WinN и OPC-сервер;

SCADA-система (supervisory control and data acquisition– система диспетчерского управления и сбора данных) – для визуализации технологических процессов и оперативного управления: AdAstra Research Group® TRACE MODE®.

Межпрограммный обмен

В настоящее время основным стандартом межпрограммного обмена данными в сфере промышленной автоматизации, безусловно, является OPC (OLE for Process Control). OPC– набор повсеместно принятых спецификаций, предоставляющих универсальный механизм обмена данными в системах контроля и управления. OPC технология обеспечивает независимость потребителей от наличия или отсутствия драйверов или протоколов, что позволяет выбирать оборудование и программное обеспечение, наиболее полно отвечающее реальным потребностям приложения.

OPC-сервер – программа, получающая данные во внутреннем формате устройства или системы и преобразующая эти данные в формат OPC. OPC-сервер является источником данных для OPC-клиентов. По своей сути OPC-сервер – это некий универсальный драйвер физического оборудования, обеспечивающий взаимодействие с любым OPC-клиентом.

В системе используется OPC-сервер CoDeSys, связанный с контроллером CoDeSys SP PLC WinNT через «общий» шлюз типа TCP/IP. Список переменных для обмена формируется в контроллере. Matlab и Trace Mode являются OPC-клиентами (рис. 33). Выбор такой конфигурации связан исключительно с простой ее настройки.

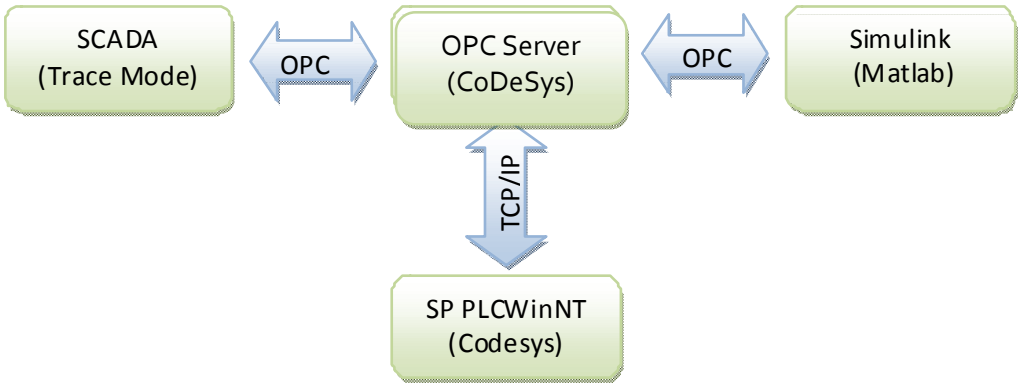


Рис. 33. Взаимодействие программ.

Запуск OPC сервера

Для запуска OPC сервера требуется выполнить следующие шаги.

1.Создатьпрограмму (проект) в среде CoDeSys с целевой платформой 3S CoDeSys SP PLC WinNT V2.4. При настройке целевой платформы следует установить параметр *Download symbol file* вкладки *General* (рис. 34).Все программы в дальнейшем будут написаны на языке ST (structured text).

2. Объявить переменные для обмена по OPC. В общем случае такие переменные могут быть объявлены в любой из программ, однако для упрощения доступа лучше все их сделать глобальными (рис. 35).

3.Написать программу PLC_PRG. В программе должен быть, по меньшей мере, один оператор, поскольку без этого она не компилируется. В нашем случае поместим в PLC_PRG те действия программы Main (см. Приложение 1), которые должны выполняться до вызовов подпрограмм (рис. 36).

4. Сохранить проект под осмысленным именем в отдельную папку.

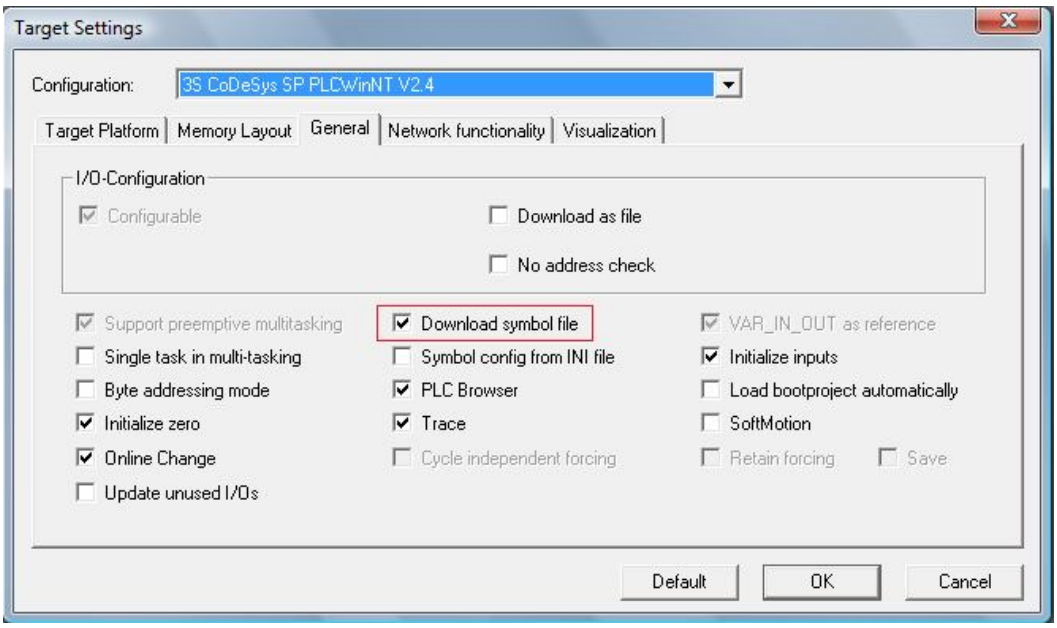


*Рис. 34.*Настройка целевой платформы.

Global_Variables

```
0001 VAR_GLOBAL
0002 (*Управление моделью*)
0003     Start:BOOL:=FALSE;
0004     Reverse:BOOL:= FALSE;
0005     LowSpeed:BOOL:=FALSE;
0006     Open:BOOL:=FALSE;
0007     Close:BOOL:=FALSE;
0008 (*Выходы модели*)
0009     Floor1:BOOL:=FALSE;
0010     Floor2:BOOL:=FALSE;
0011     Floor3:BOOL:=FALSE;
0012     Floor4:BOOL:=FALSE;
0013     UpFloor2:BOOL:=FALSE;
0014     UpFloor3:BOOL:=FALSE;
0015     UpFloor4:BOOL:=FALSE;
0016     DownFloor1:BOOL:=FALSE;
0017     DownFloor2:BOOL:=FALSE;
0018     DownFloor3:BOOL:=FALSE;
0019     Opened:BOOL:=FALSE;
0020     Closed:BOOL:=FALSE;
0021     y:REAL:=10;
0022     x:REAL:=0;
0023 (*Связь с Trace Mode*)
0024     CallsUp:BYTE:=0;
0025     CallsDown:BYTE:=0;
0026     Orders:BYTE:=0;
0027     CallsUpForm:BYTE:=0;
0028     CallsDownForm:BYTE:=0;
0029     OrdersForm:BYTE:=0;
0030 (*"Внутренние" переменные*)
0031     Floors:BYTE:=0;
0032     Appr_Floors_Up:BYTE:=0;
0033     Appr_Floors_Down:BYTE:=0;
0034     State:BYTE:=1;
0035 END_VAR
```

Рис.35. Объявление глобальных переменных

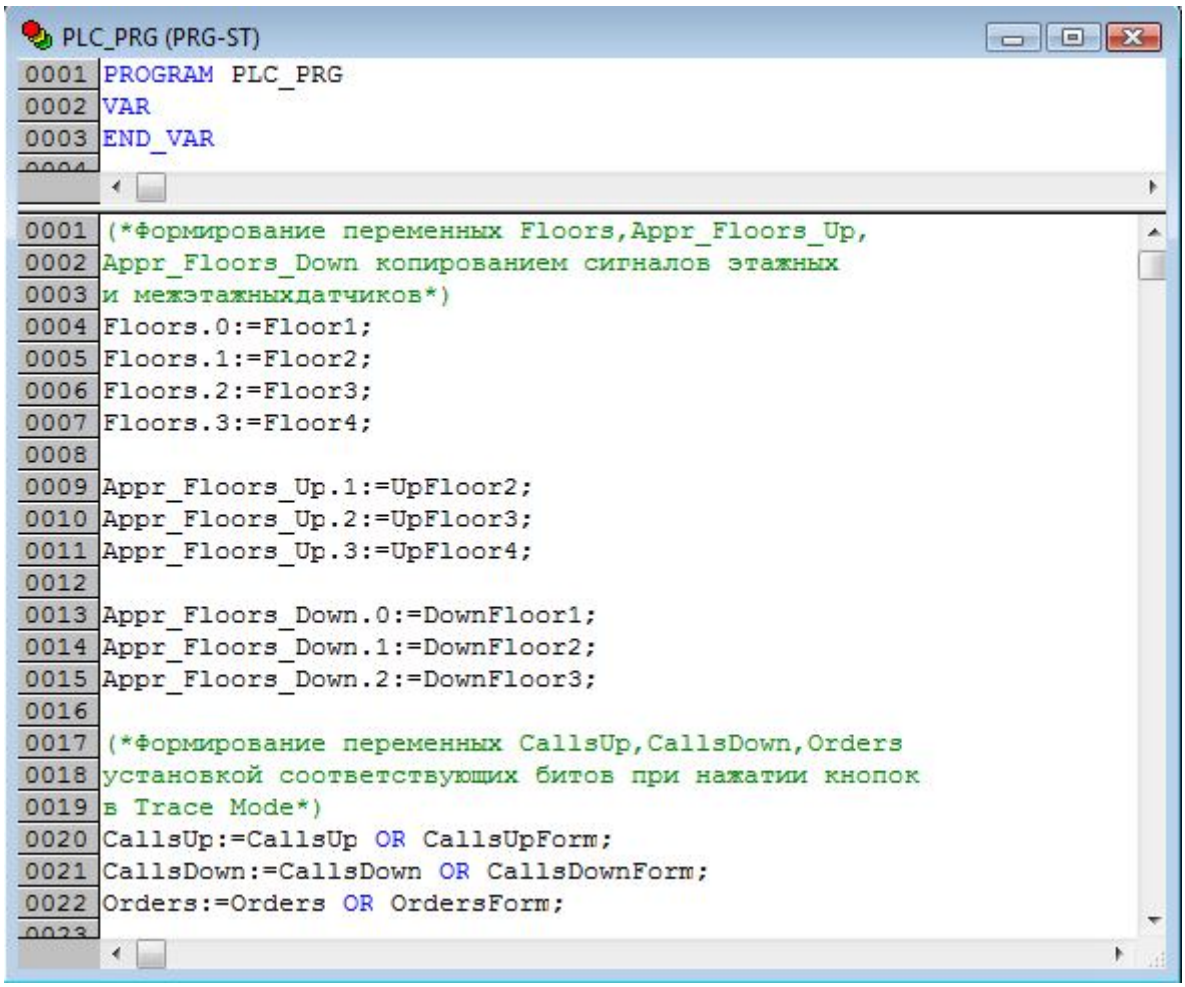


*Рис. 36.*Программа PLC_PRG (не закончена).

5. Запустить PLC WinNT (*Пуск→ Все программы → 3S Software→CoDeSys SP PLC WinNT→CoDeSys SP PLC WinNT V2.4*) (рис. 37), установить связь и загрузить программу в «контроллер».

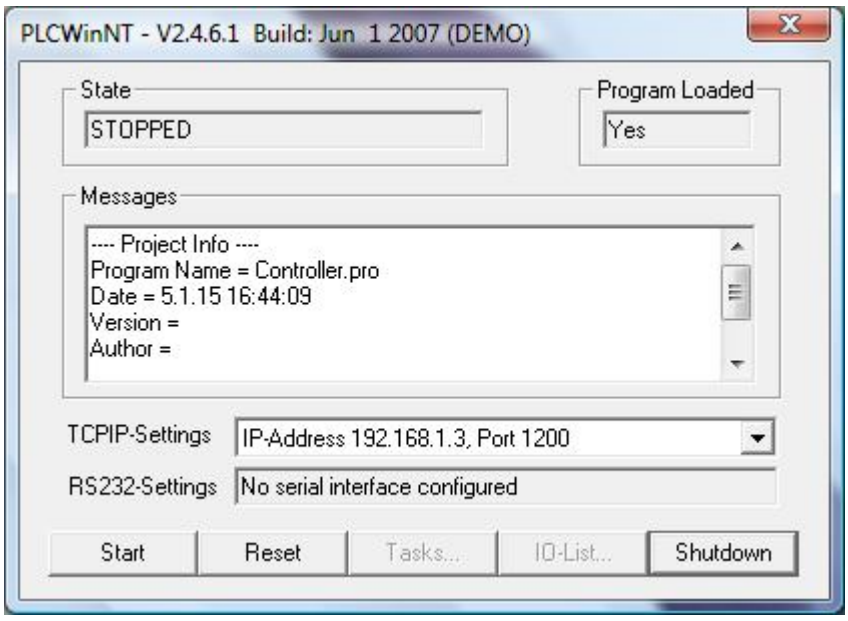


Рис. 37. Окно программы PLC WinNT.

6. Отключить CoDeSys от PLC WinNT и перейти в Опции (Options) в меню Project. Выбрать Symbol Configuration и установить галочку*Dump symbol entries* (рис. 38).

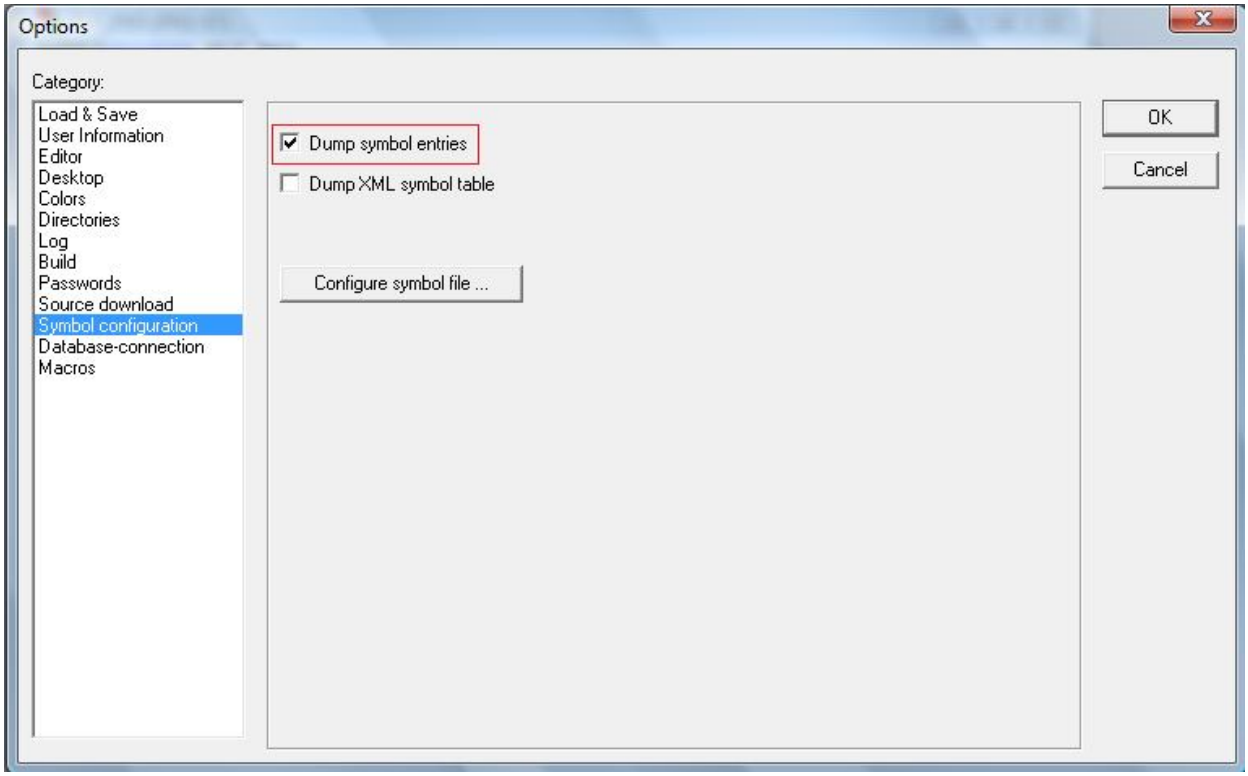


*Рис. 38.*Установка опций.

7. Сконфигурировать «символьный файл», выбрав переменные для обмена по OPC (рис.39). В нашем случае это будут все глобальные переменные за исключением «внутренних» Floors, Appr_Floors_Up, Appr_Floors_Down, State.

8. Настроить параметры OPC сервера. Для этого необходимо запустить конфигуратор OPCсервера (*Пуск → Все программы → 3S Software →Communication → CoDeSys OPC Configurator*). В окне конфигуратора требуется добавить PLC (*Append PLC*) и настроить соединение (*Connection*). По существу достаточно выбрать из списка соединение, настроенное в проекте CoDeSys (рис. 40).

9. «Перестроить» программу (проект) CoDeSys, включив в нее все изменения, сделанные после загрузки в контроллер. Для этого требуется вызвать команду *Clean all* из меню *Project*, затем команду *Rebuild all* из того же меню. CoDeSys перекомпилирует программу и перезагрузит проект при следующем подключении к ПЛК.

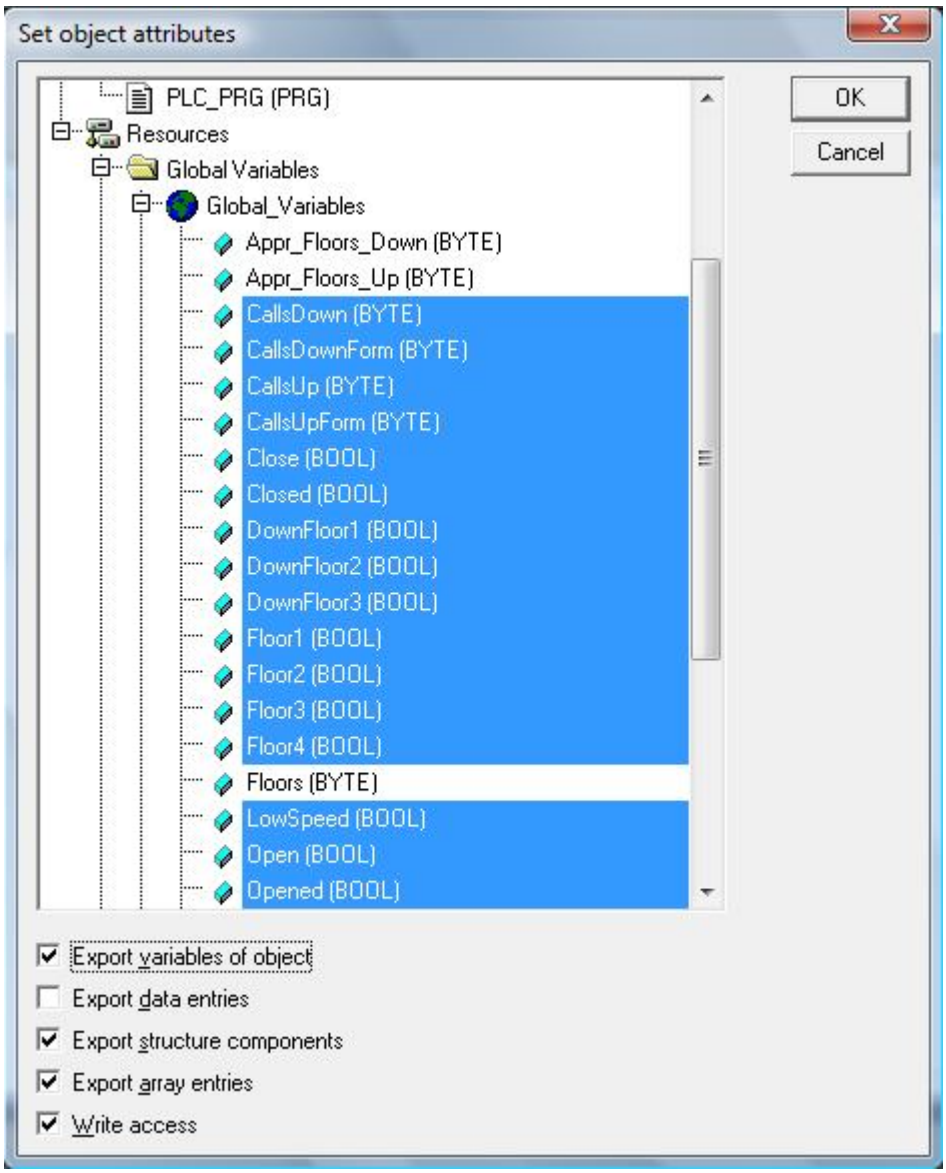


*Рис. 39.*Выбор переменных для обмена по OPC.

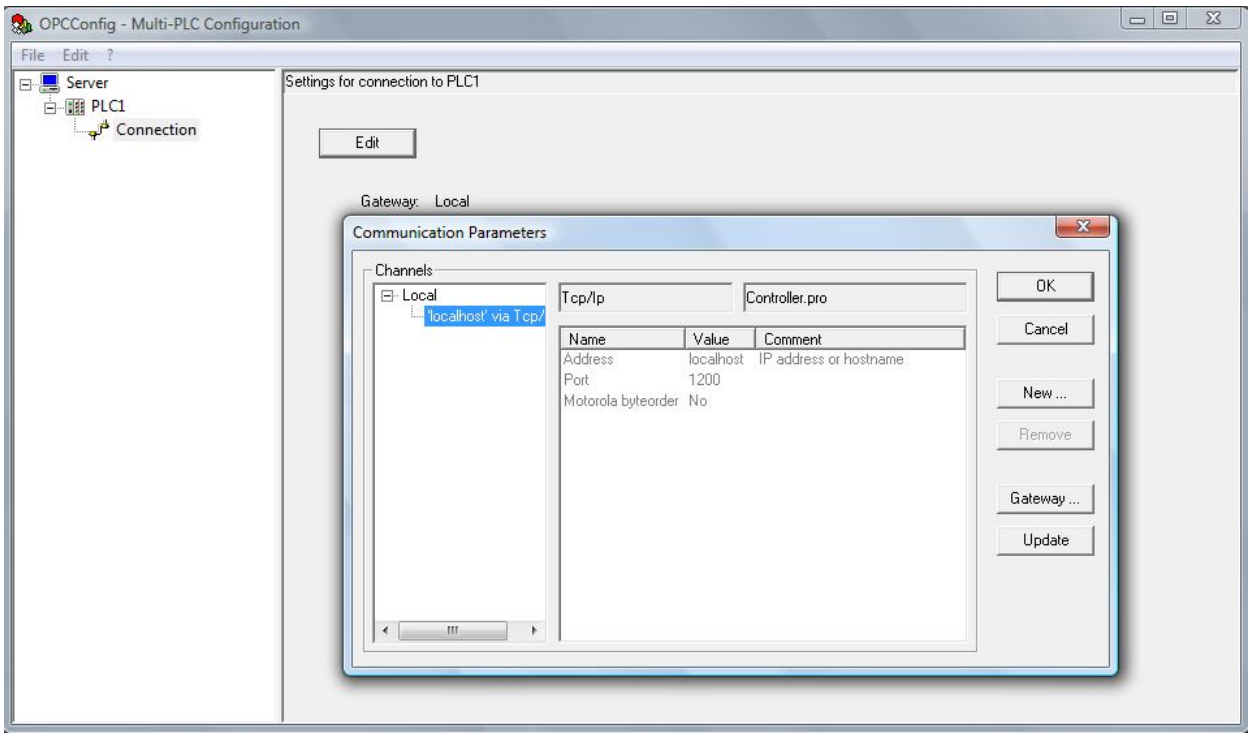


*Рис. 40.*Настройка OPC сервера.

Имитационная модель лифтового механизма

Модель строится в системе Simulink (рис. 41). Задача модели состоит в определении положения кабины в шахте, положения дверей, состояния этажных и межэтажных датчиков.

Входами модели являются:

команды закрытия и открытия для привода дверей;

команды пуска, реверса и перехода на пониженную скорость для главного привода кабины.

Все команды формируются контроллером и передаются в модель через блок OPC Read.

Подсистема Main drive (рис. 42) представляет собой модель главного привода, включающего преобразователь частоты, двигатель, редуктор и другие элементы, приводящие в движение кабину лифта. Модель построена на двух интеграторах, на выходе первого из которых формируется вертикальная скорость кабины, на выходе второго – ее положение в шахте. Релейный регулятор скорости обеспечивает постоянное ускорение разгона и торможения, имитируя тем самым реальное поведение преобразователя частоты. Задание по скорости формируется с помощью трех переключателей, управляемых входными сигналами блока. На рис. 42 показаны также окна настроек релейных элементов и интеграторов.

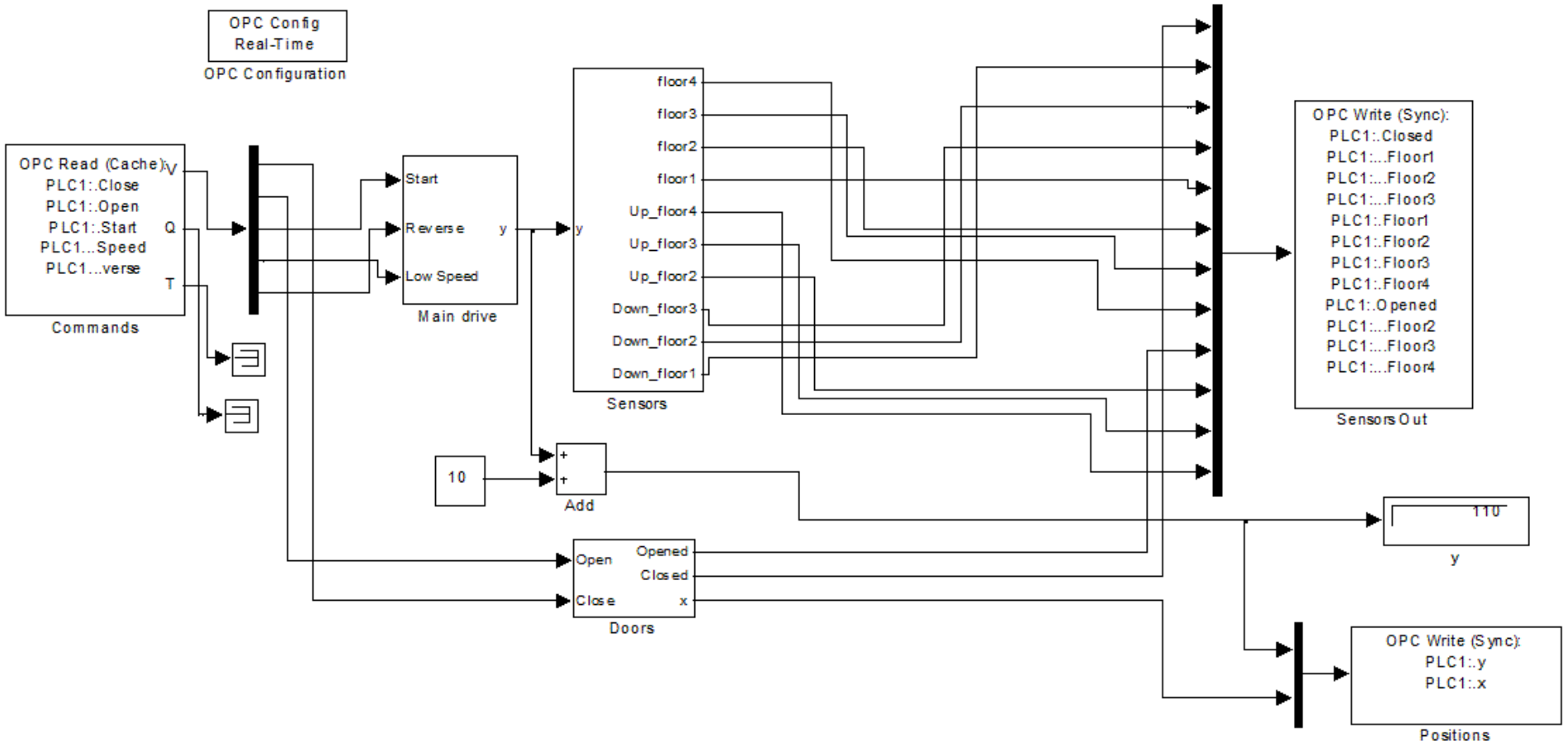


Рис. 41. Simulink-модель лифтового механизма.

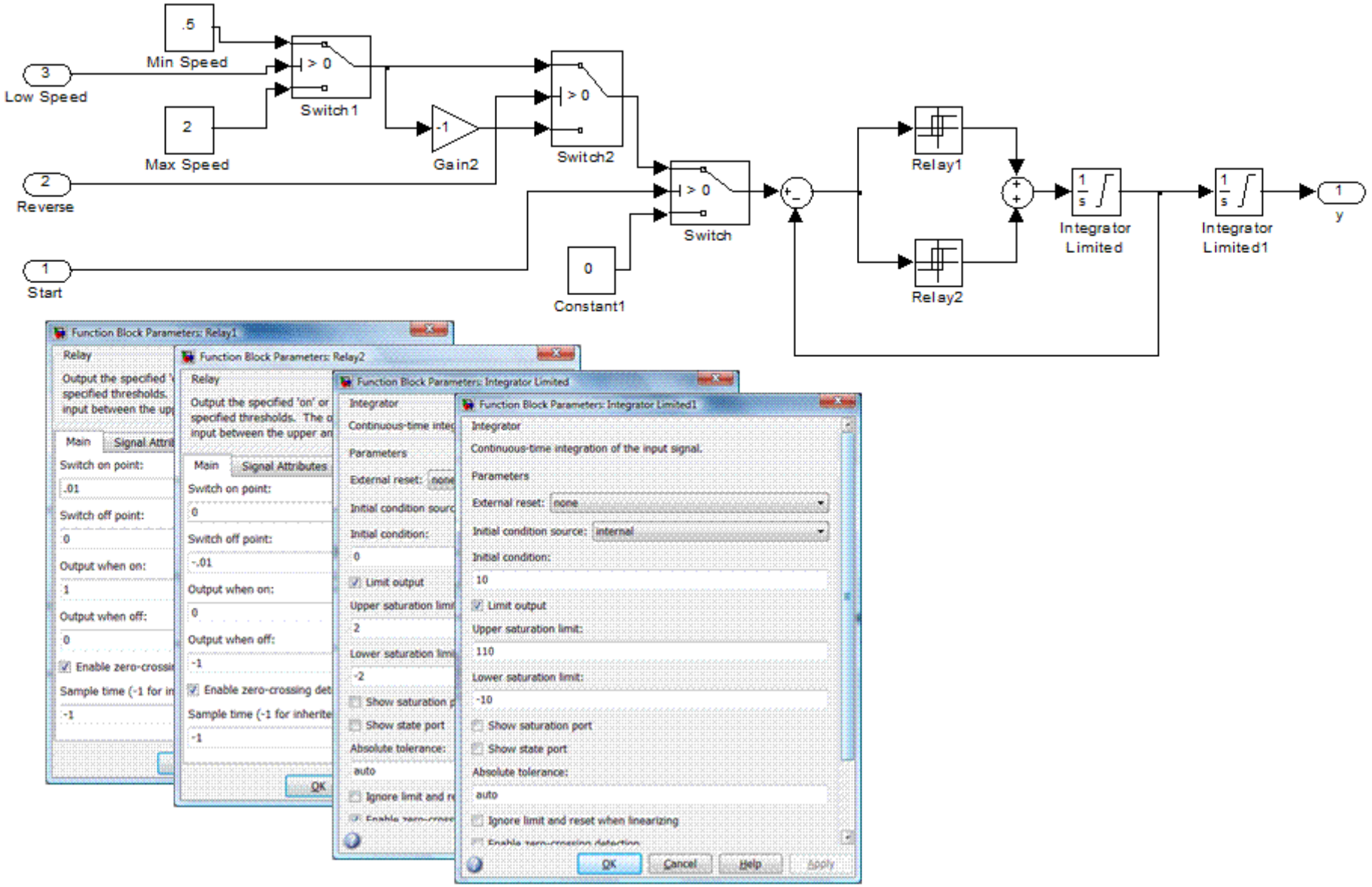


Рис. 42. Подсистема Main Drive и ее настройки.

На рис. 43. показана подсистема Sensors, формирующая модельные эквиваленты сигналов этажных и межэтажных датчиков.

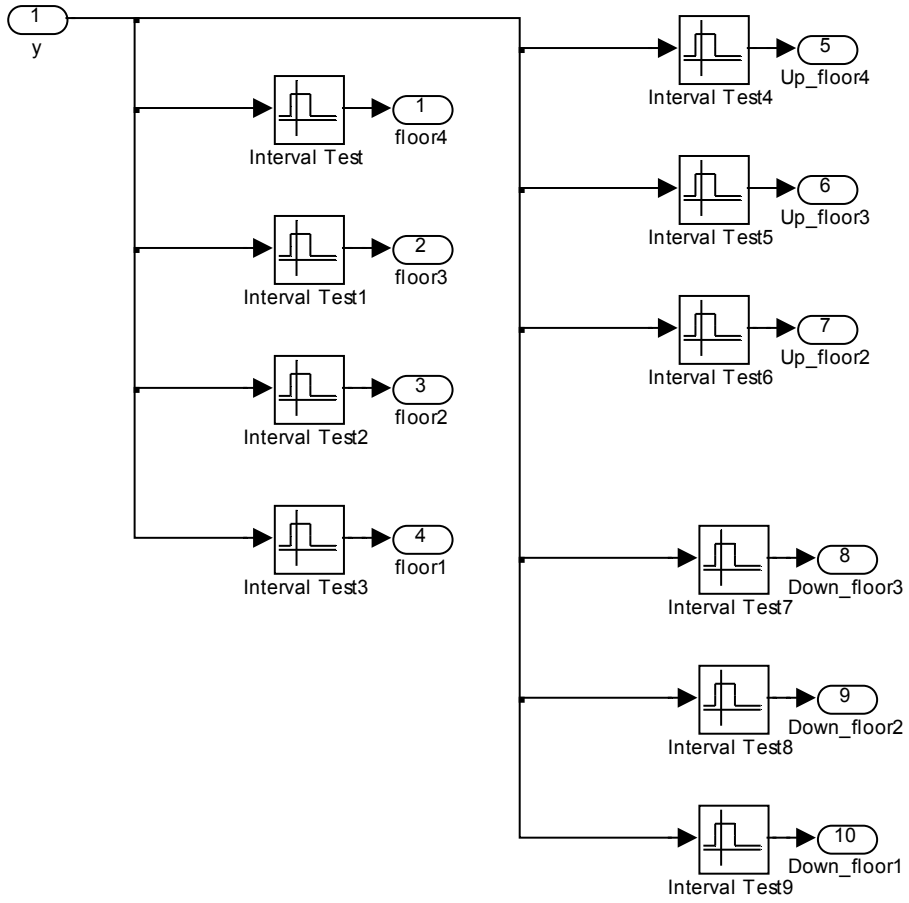


Рис. 43. Подсистема Sensors.

Подсистема построена на блоках Interval Test, фиксирующих нахождение входного сигнала (положения кабины) в заданном диапазоне. Пределы диапазонов блоков приведены в табл. 11.

Таблица 11

Параметры блоков Interval Test

Датчик	UpperLimit	Lower Limit
Interval Test, floor4	101	99
Interval Test1, floor3	67	65
Interval Test2, floor2	34	32
Interval Test3, floor1	1	-1
Interval Test4, Up_floor4	96	94
Interval Test5, Up_floor3	62	60
Interval Test6, Up_floor2	29	27
Interval Test7, Down_floor3	71	69
Interval Test8, Down_floor2	39	37
Interval Test9, Down_floor1	6	4

Подсистема Doors представлена на рис. 44. Она моделирует работу привода дверей и формирует сигналы положения дверей и состояния концевых выключателей.

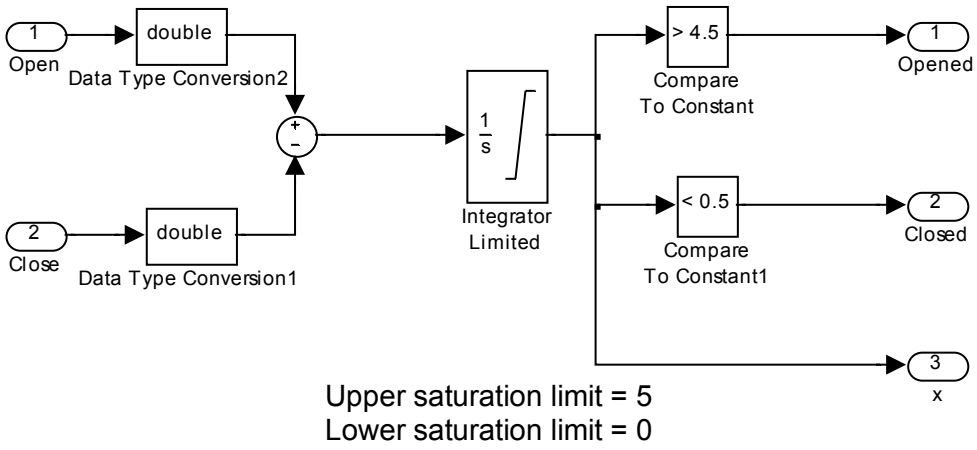


*Рис. 44.*Подсистема Doors.

Для создания блоков OPC Read и OPC Write рекомендуется использовать утилиту opctool, вызвав ее из окна команд Matlab. Эта утилита позволяет достаточно просто подключиться к OPC серверу и сконфигурировать OPC клиента, рис. 45, 46.

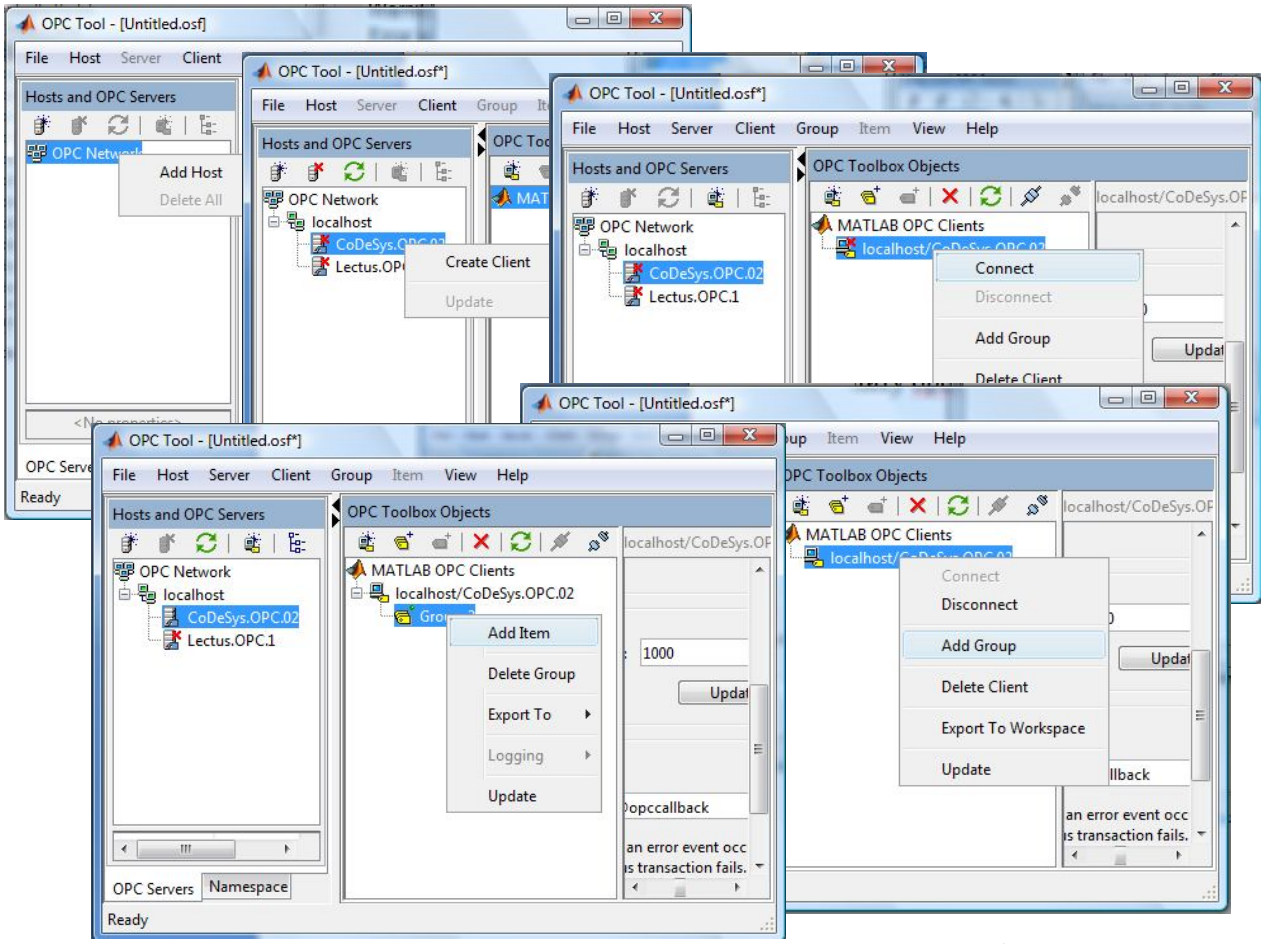


*Рис. 45.*Подключение к OPC-серверу в opctool.

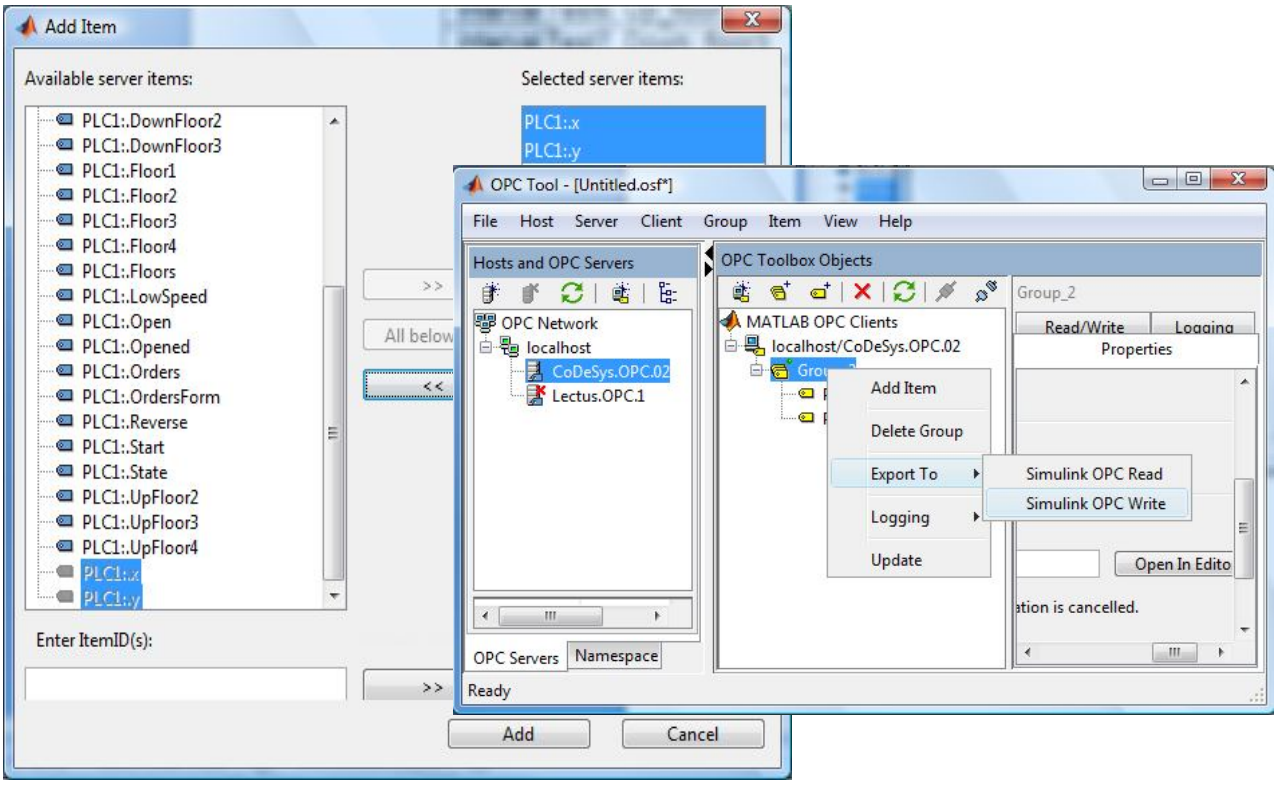


*Рис. 46.*Выбор тегов и создание блока OPC Write (Read).

Последовательность действий по созданию блоков OPC Read и OPC Write следующая, см. рис. 45:

1) Add Host – добавить узел сети, на котором запущен OPC сервер. В нашем случае сервер запускается на локальном компьютере, поэтому выбирается localhost. После выбора узла появляется список доступных OPC серверов;

2) Create Client – создать клиента, здесь выбирается конкретный OPC сервер, в нашем случае CoDeSys;

3) Connect – подключиться к серверу (в частности для получения информации о тегах-переменных);

4) Add Group – добавить группу тегов (в OPC обмен всегда ведется группами, группы создает клиент);

5) Add Item – добавить теги. Из списка доступных тегов выбираются те, что должны образовать группу, после этого они становятся доступны в окне OPC Toolbox Object;

6) Export To – экспортировать в Simulink OPC Read или OPC Write. Созданный блок попадает в новую Simulink модель, откуда его можно скопировать.

После создания всех необходимых блоков OPC Read и OPC Write утилиту opctool можно закрыть (с сохранением или без сохранения конфигурации). Связав интерфейсные блоки с вычислительной частью, запустите модель на исполнение, задав время расчета достаточно большим. Блок OPC Config будет создан автоматически.

Разработка управляющей программы

Программа для виртуального контроллера написана на языке ST в CoDeSys. Она практически полностью реализует логику управления и алгорит-

мы из Приложения №1. Исключение составляет подпрограмма Doors_Closing: в ней отсутствует обработка сигнала о заклинивании дверей. Этот сигнал, как и ситуация с заклиниванием, в данной модели не предусмотрены в целях упрощения.

Глобальные переменные были объявлены ранее, см. рис. 35. Главная программа PLC_PRG дополнена вызовом подпрограмм, рис. 47.

PLC_PRG (PRG-ST)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 END_VAR
```

```
0001 (*Формирование переменных Floors,Appr_Floors_Up,
0002 Appr_Floors_Down копированием сигналов этажных
0003 и межэтажныхдатчиков*)
0004 Floors.0:=Floor1;
0005 Floors.1:=Floor2;
0006 Floors.2:=Floor3;
0007 Floors.3:=Floor4;
0008
0009 Appr_Floors_Up.1:=UpFloor2;
0010 Appr_Floors_Up.2:=UpFloor3;
0011 Appr_Floors_Up.3:=UpFloor4;
0012
0013 Appr_Floors_Down.0:=DownFloor1;
0014 Appr_Floors_Down.1:=DownFloor2;
0015 Appr_Floors_Down.2:=DownFloor3;
0016
0017 (*Формирование переменных CallsUp,CallsDown,Orders
0018 установкой соответствующих битов при нажатии кнопок
0019 в Trace Mode*)
0020 CallsUp:=CallsUp OR CallsUpForm;
0021 CallsDown:=CallsDown OR CallsDownForm;
0022 Orders:=Orders OR OrdersForm;
0023
0024 CASE State OF
0025 1: Nearest_Lower_Floor;
0026 2: Direction_Choose;
0027 3: Move_Up;
0028 4: Move_Down;
0029 5: Doors_Openning;
0030 6: Doors_Closing;
0031 END_CASE;
```

*Рис. 47.*Программа PLC_PRG.

Подпрограммы представлены на рис. 48-53.

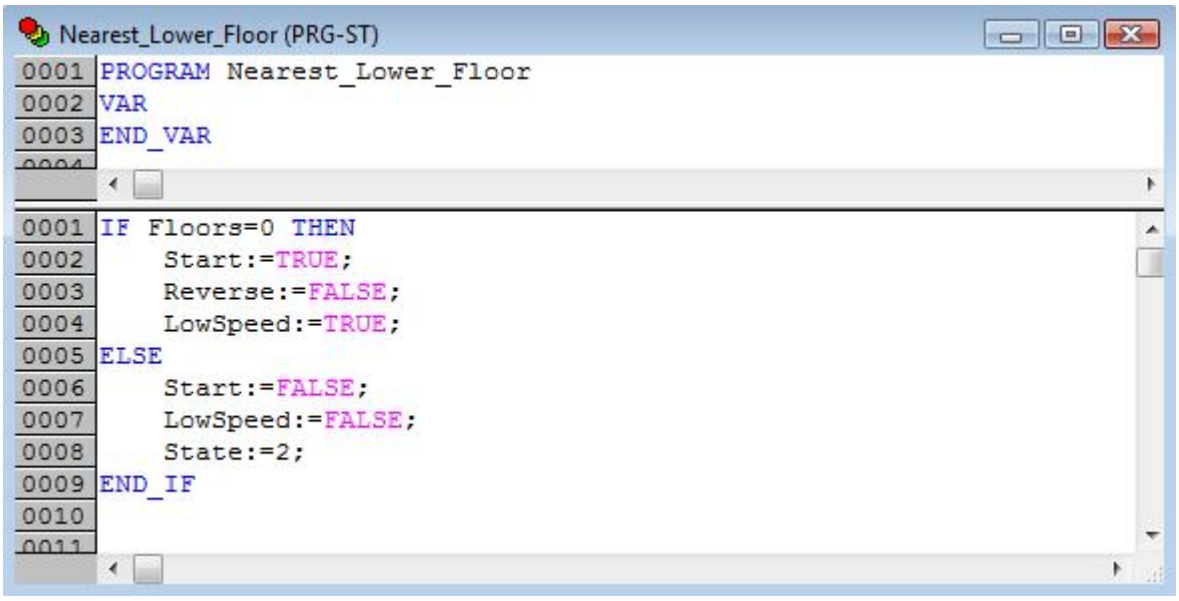


*Рис. 48.*Подпрограмма Nearest_Lower_Floor.

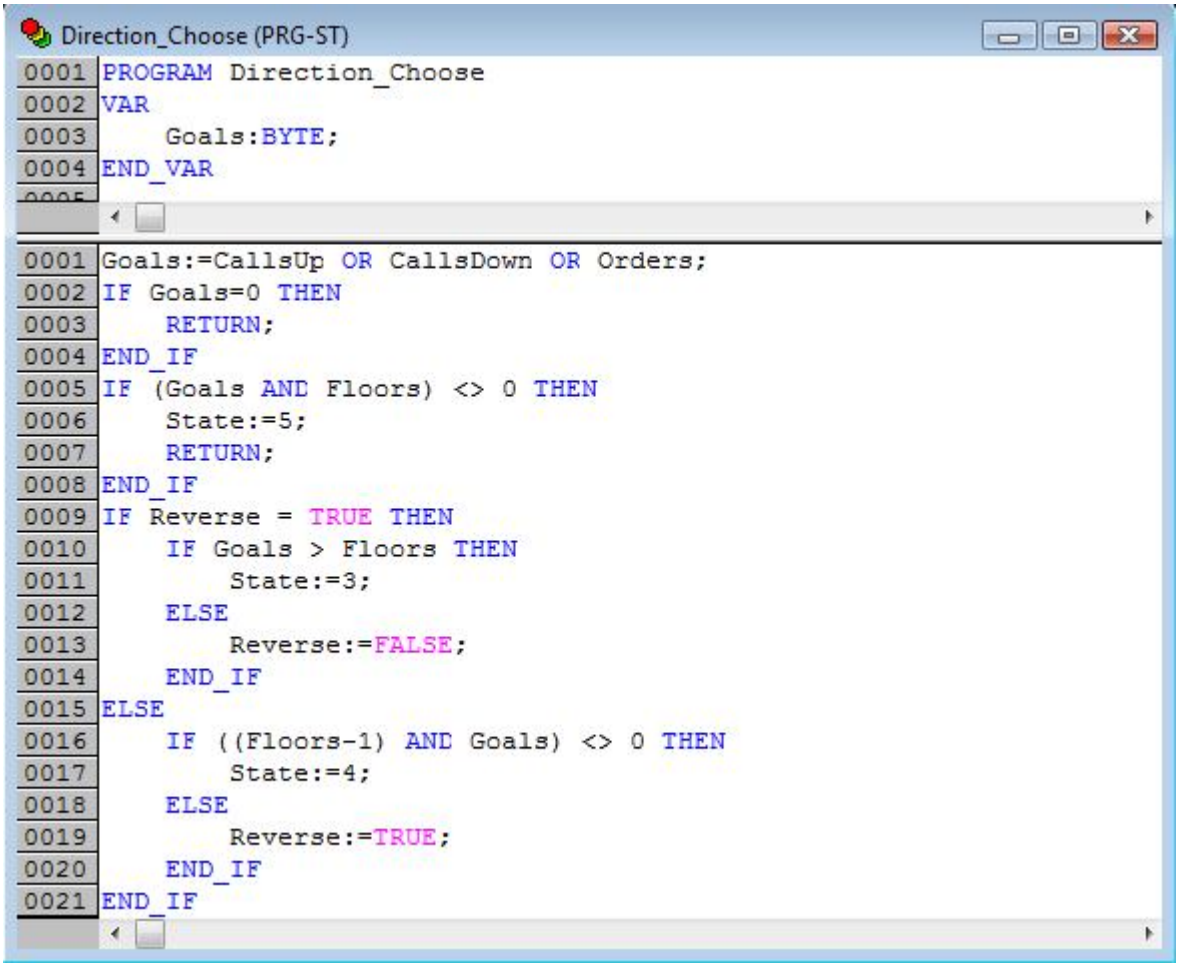


*Рис. 49.*Подпрограмма Direction_Choose

Move_Up (PRG-ST)

```
0001 PROGRAM Move_Up
0002 VAR
0003     Goals:BYTE;
0004 END_VAR
```

```
0001 Goals:=CallsUp OR CallsDown OR Orders;
0002 IF (Floors <> 0) AND LowSpeed THEN
0003     Start:=FALSE;
0004     LowSpeed:=FALSE;
0005     State:=5;
0006 ELSE
0007     Start:=TRUE;
0008     IF ((Appr_Floors_Up AND Orders) <> 0) OR
0009        ((Appr_Floors_Up AND CallsUp) <> 0) THEN
0010         LowSpeed:=TRUE;
0011     ELSIF ((Appr_Floors_Up AND CallsDown) <> 0) AND
0012           ((Appr_Floors_Up AND CallsDown) >
0013            (Appr_Floors_Up XOR Goals)) THEN
0014         LowSpeed:=TRUE;
0015     END_IF
```

*Рис. 50.*Подпрограмма Move_Up.

Move_Down (PRG-ST)

```
0001 PROGRAM Move_Down
0002 VAR
0003     Goals:BYTE;
0004 END_VAR
```

```
0001 Goals:=CallsUp OR CallsDown OR Orders;
0002
0003 IF (Floors <> 0) AND LowSpeed THEN
0004     Start:=FALSE;
0005     LowSpeed:=FALSE;
0006     State:=5;
0007 ELSE
0008     Start:=TRUE;
0009     IF ((Appr_Floors_Down AND Orders) <> 0) OR
0010        ((Appr_Floors_Down AND CallsDown) <> 0) THEN
0011         LowSpeed:=TRUE;
0012     ELSIF ((Appr_Floors_Down AND CallsUp) <> 0) AND
0013           (((Appr_Floors_Down - 1) AND  Goals) = 0) THEN
0014         LowSpeed:=TRUE;
0015     END_IF
0016 END_IF
0017
0018
0019
0020
```

*Рис. 51.*Подпрограмма Move_Down.

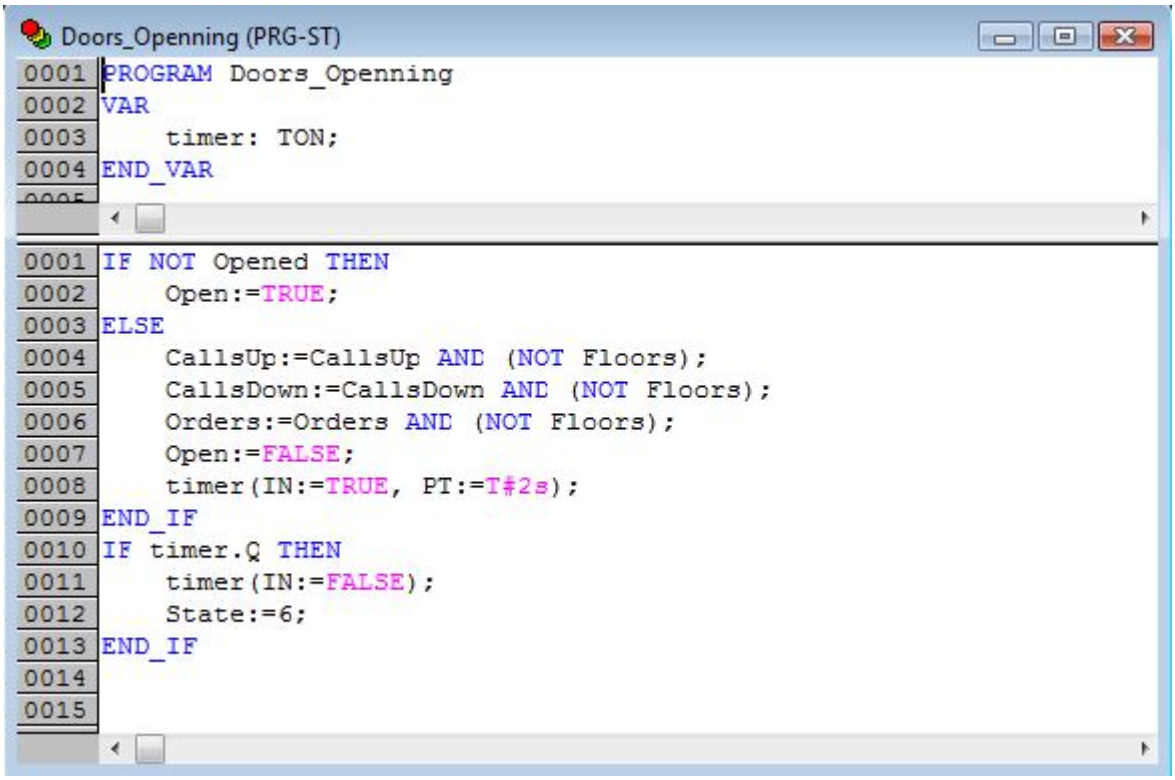


*Рис. 52.*Подпрограмма Doors_Opening.

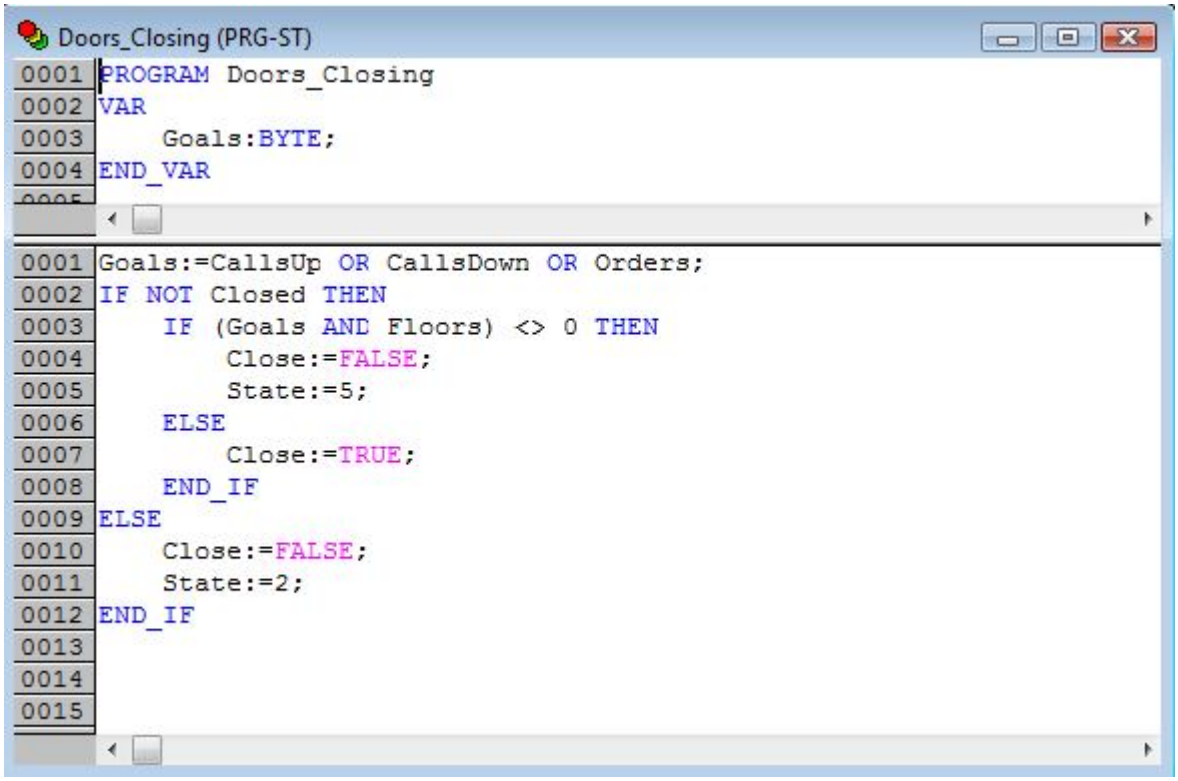


*Рис. 53.*Подпрограмма Doors_Closing.

Визуализация системы

Задачами визуализации являются отображение положения кабины и ее дверей и имитация этажных панелей вызовов и панели приказов в кабине.

Экран визуализации и необходимые каналы ввода-вывода созданы в системе Trace Mode 6.

На рис. 54 показано создание каналов Источники/Приемники посредством которых будет осуществляться обмен с OPC сервером.

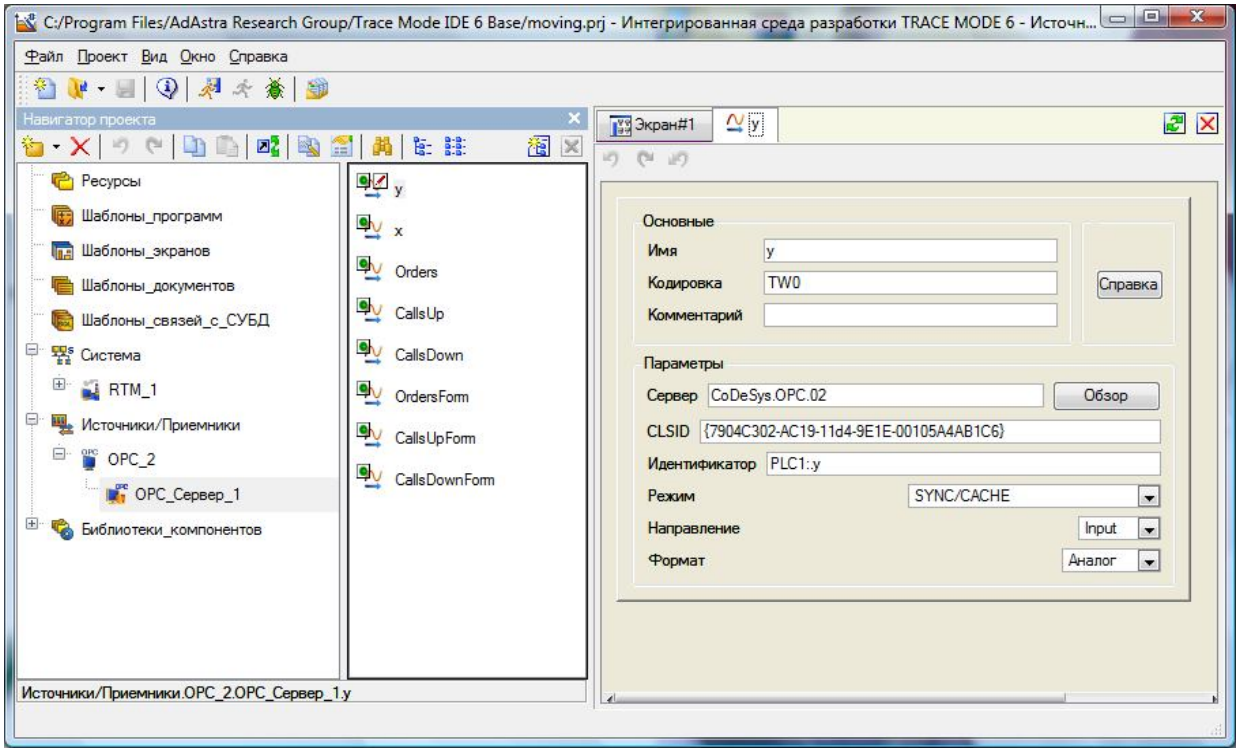


*Рис. 54.*Создание каналов Источники/Приемники.

Как и любая SCADA-система, Trace Mode является «естественным» OPC клиентом, поэтому настройка каналов ввода-вывода осуществляется просто. В Навигаторе проекта в разделе «Источники/Приемники» последовательно создаются группы «OPC», «OPC сервер» и компоненты «OPC».В свойствах компонента с помощью кнопки «Обзор» запускается Браузер OPC, который выводит список доступных OPC серверов. После выбора сервера становится доступным список его переменных, см. рис. 55,из которого и выбирается необходимая переменная. Единственным параметром настройки, требующим уточнения, является «Направление»: требуется указать направление передачи данных (Input– чтение из сервера, Output – запись в сервер). В нашем случае переменные y, x, Orders, CallsUp, CallsDown будут входными, OrdersForm, CallsUpForm, CallsDownForm – выходными.

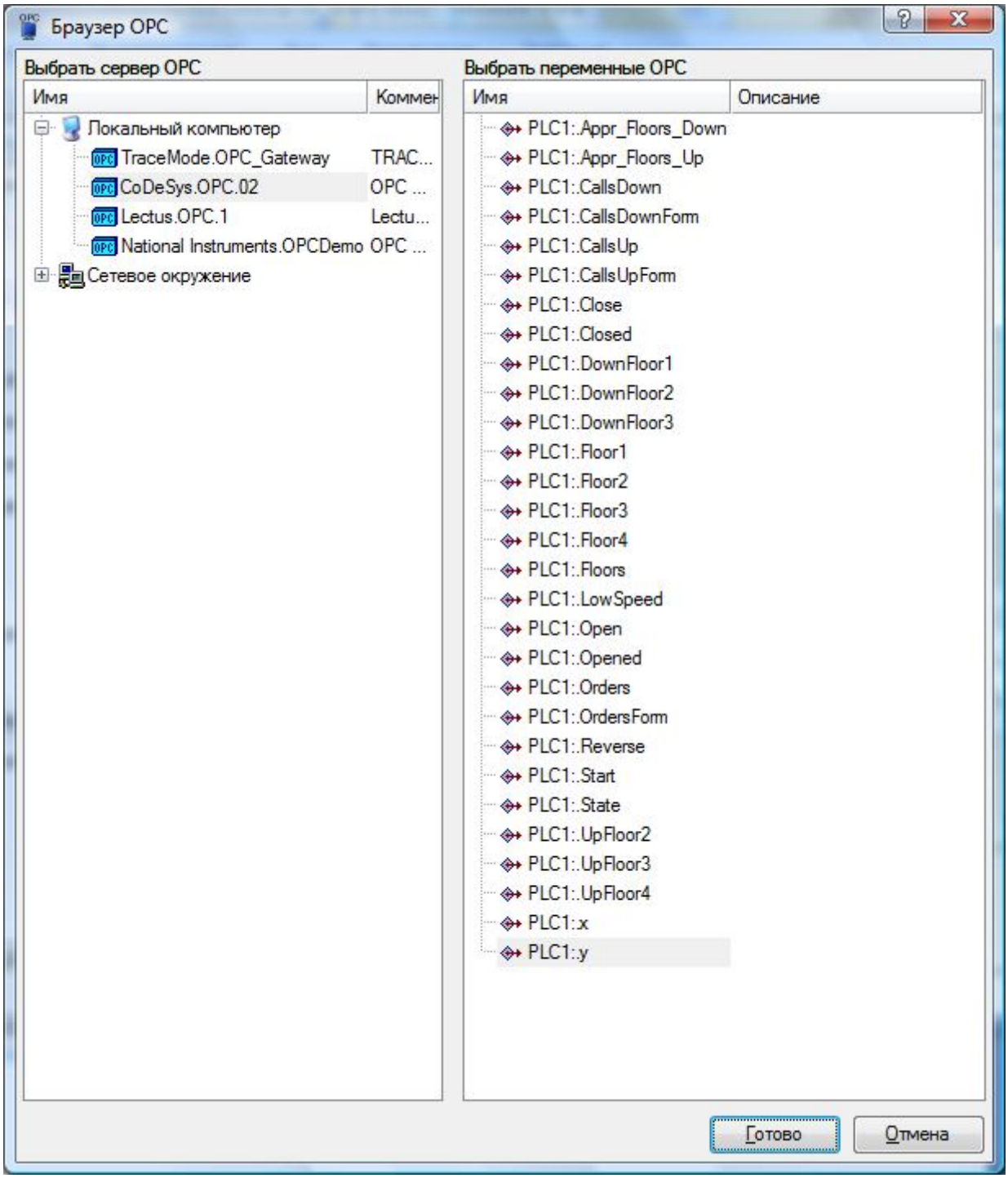
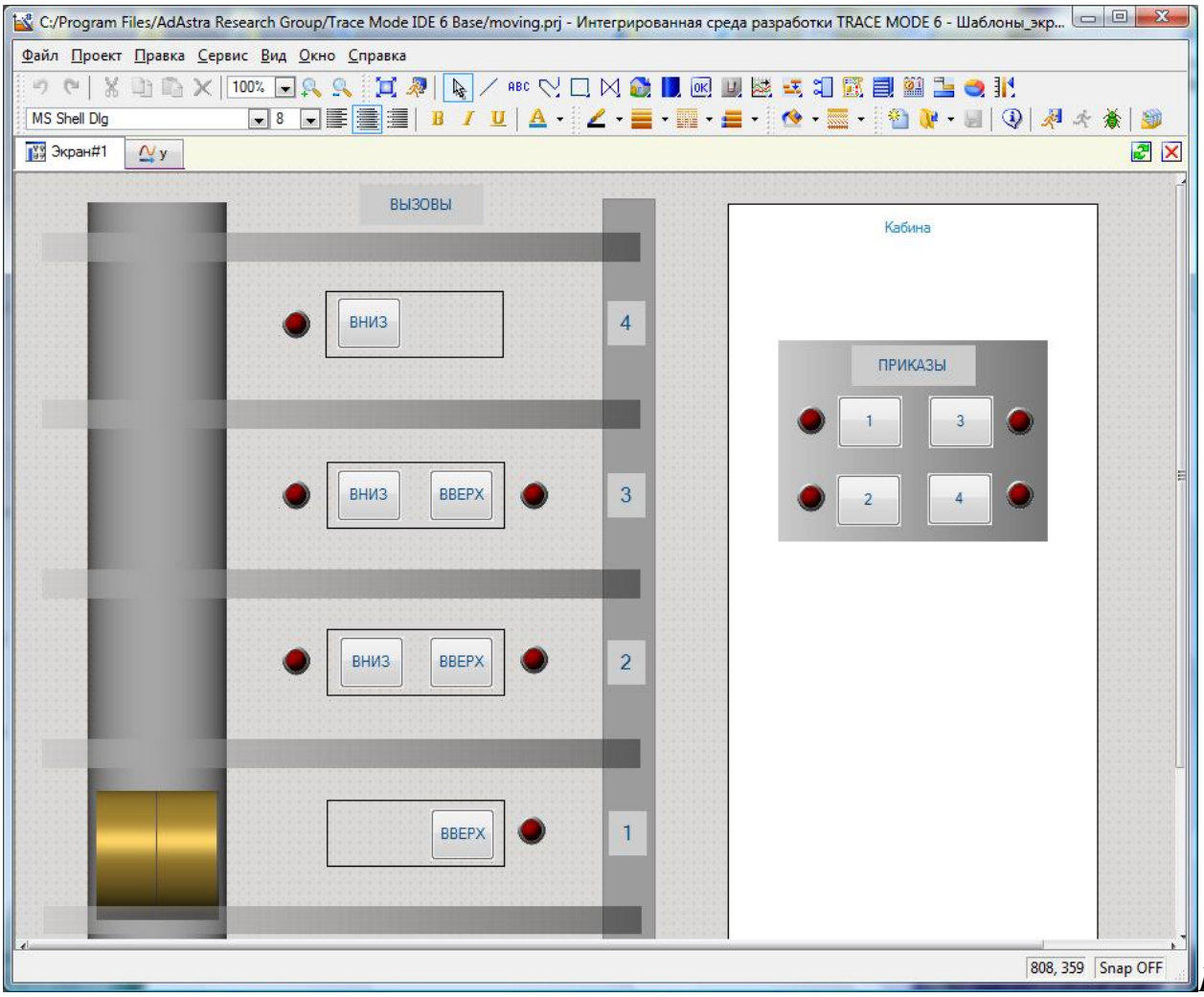


Рис. 55. Выбор переменные в Браузере OPC.

Экран визуализации показан на рис. 56. Экран содержит схематичные изображения шахты с кабиной лифта, этажных панелей вызовов и панели приказов в кабине с кнопками и индикаторами.

Кабина перемещается в шахте, принимая положение, вычисленное имитационной моделью. Двери лифта «открываются» и «закрываются».

С помощью кнопок формируются вызовы и приказы. Индикаторы фиксируют еще неотработанные системой вызовы и приказы.



*ис. 56.*Экран визуализации.

Аргументы экрана с привязками показаны на рис. 57.

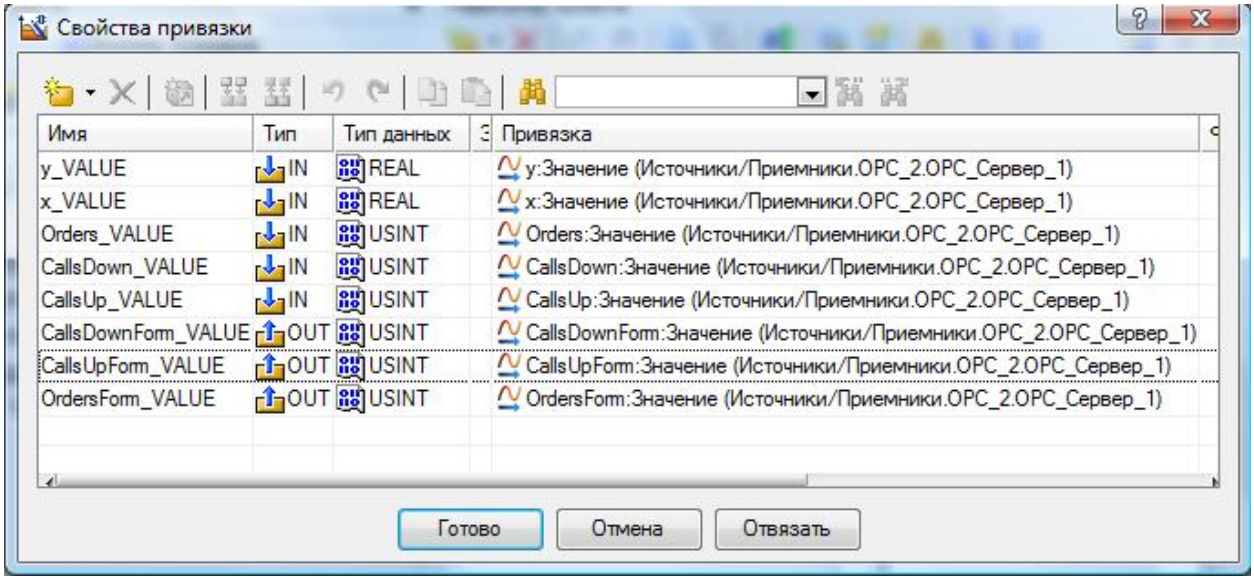


*Рис. 57.*Аргументы экрана.

Движение кабины вдоль шахты происходит под управлением аргумента экрана y_VALUE, привязанной к OPC переменной y. На рис. 58 показано формирование траектории для левой половины дверей.

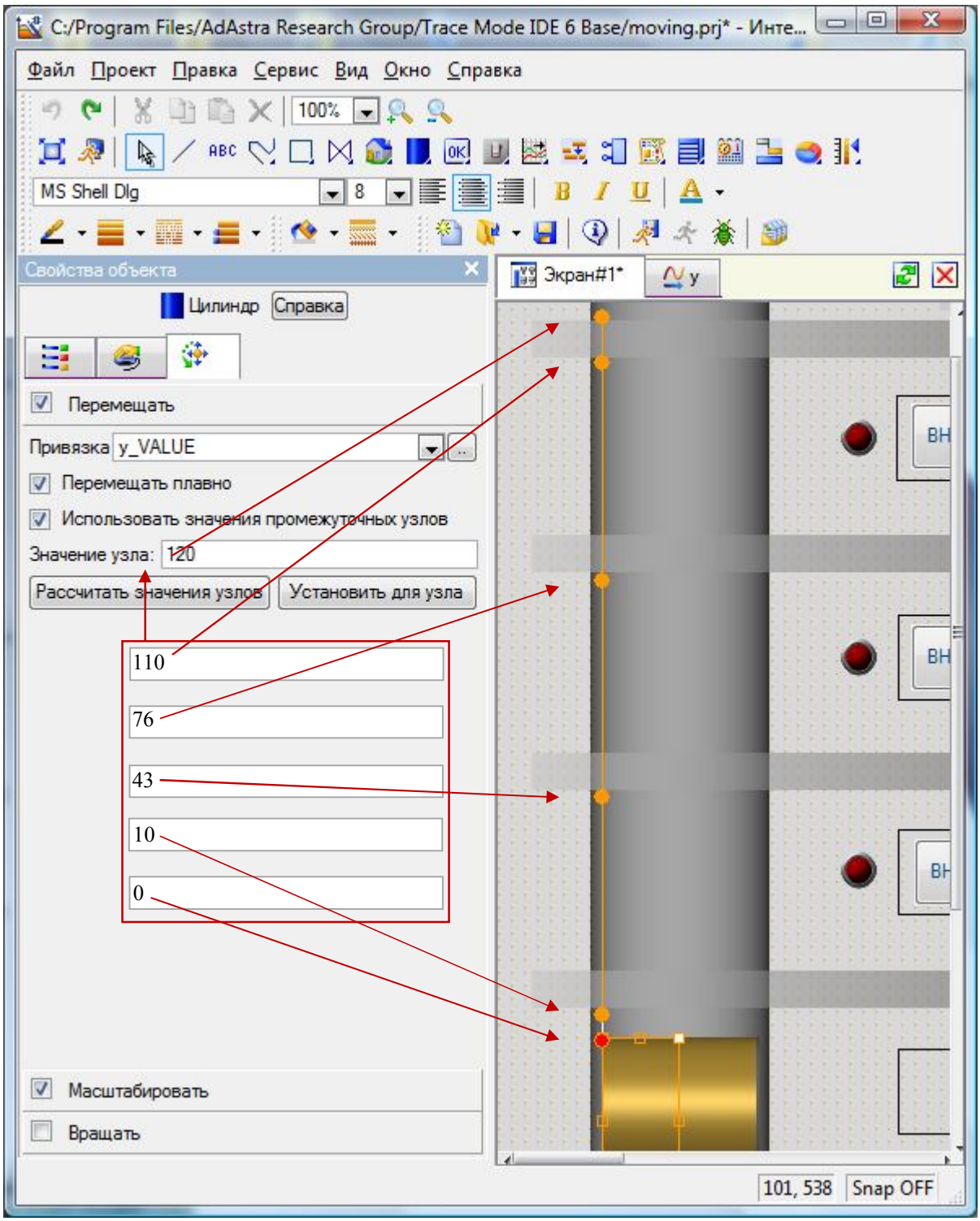


Рис. 58. Формирование траекторий движения левой половины дверей.

К сожалению, в Trace Mode 6 отсутствует возможность группирования графических элементов, поэтому траектории движения согласованно двигающихся объектов нужно задавать для каждого из них в отдельности. Практически проще сначала создать и полностью настроить один графический объект, потом сделать и отредактировать его копии. Именно таким способом и были сформированы правая половина дверей и рамка обрамления кабины (она становится видимой при открытых дверях).

Открытие дверей имитируется горизонтальным масштабированием «половинок» под управлением аргумента экрана x_VALUE, привязанной к OPC переменной x. На рис. 59показана настройка масштабирования для левой половины дверей.

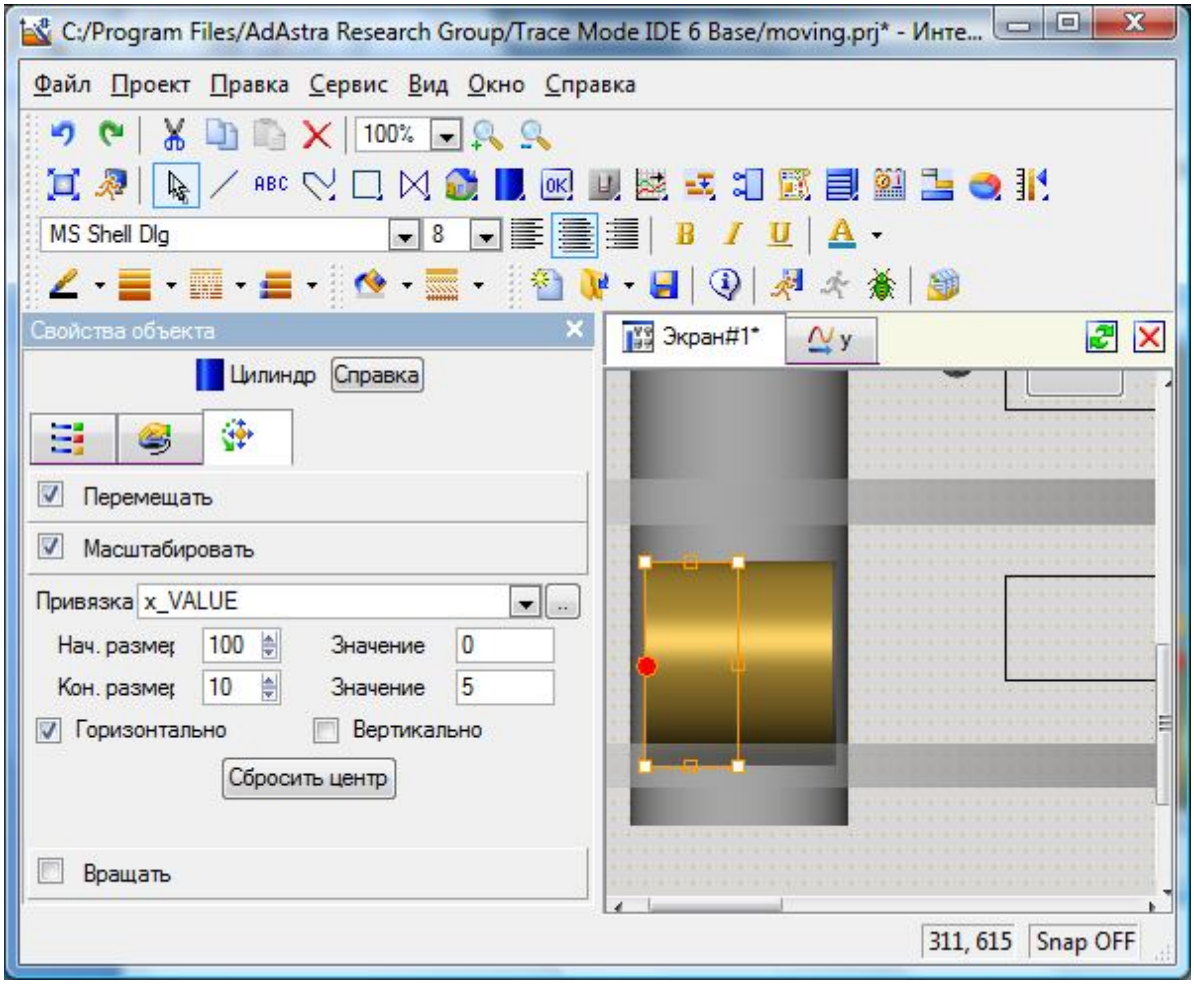


Рис.59. Настройка «открытия» левой половины дверей.

Настройка «открытия» правой половины дверей аналогична за исключением положения центра масштабирования.

Настройка индикаторов неотработанных вызовов и приказов демонстрируется на рис. 60. на примере индикатора вызова вверх с первого этажа. Другие индикаторы настраиваются аналогично (табл. 12).

Таблица 12

Настройка индикаторов вызовов и приказов

Индикатор	Привязка	Константа
Вызов вверх 1 этаж	CallsUp_VALUE	0x1
Вызов вверх 2 этаж	CallsUp_VALUE	0x2
Вызов вверх 3 этаж	CallsUp_VALUE	0x4
Вызов вниз 2 этаж	CallsDown_VALUE	0x2
Вызов вниз 3 этаж	CallsDown_VALUE	0x4

Вызов вниз 4 этаж	CallsDown_VALUE	0x8
Приказ 1 этаж	Orders_VALUE	0x1
Приказ 2 этаж	Orders_VALUE	0x2
Приказ 3 этаж	Orders_VALUE	0x4
Приказ 4 этаж	Orders_VALUE	0x8

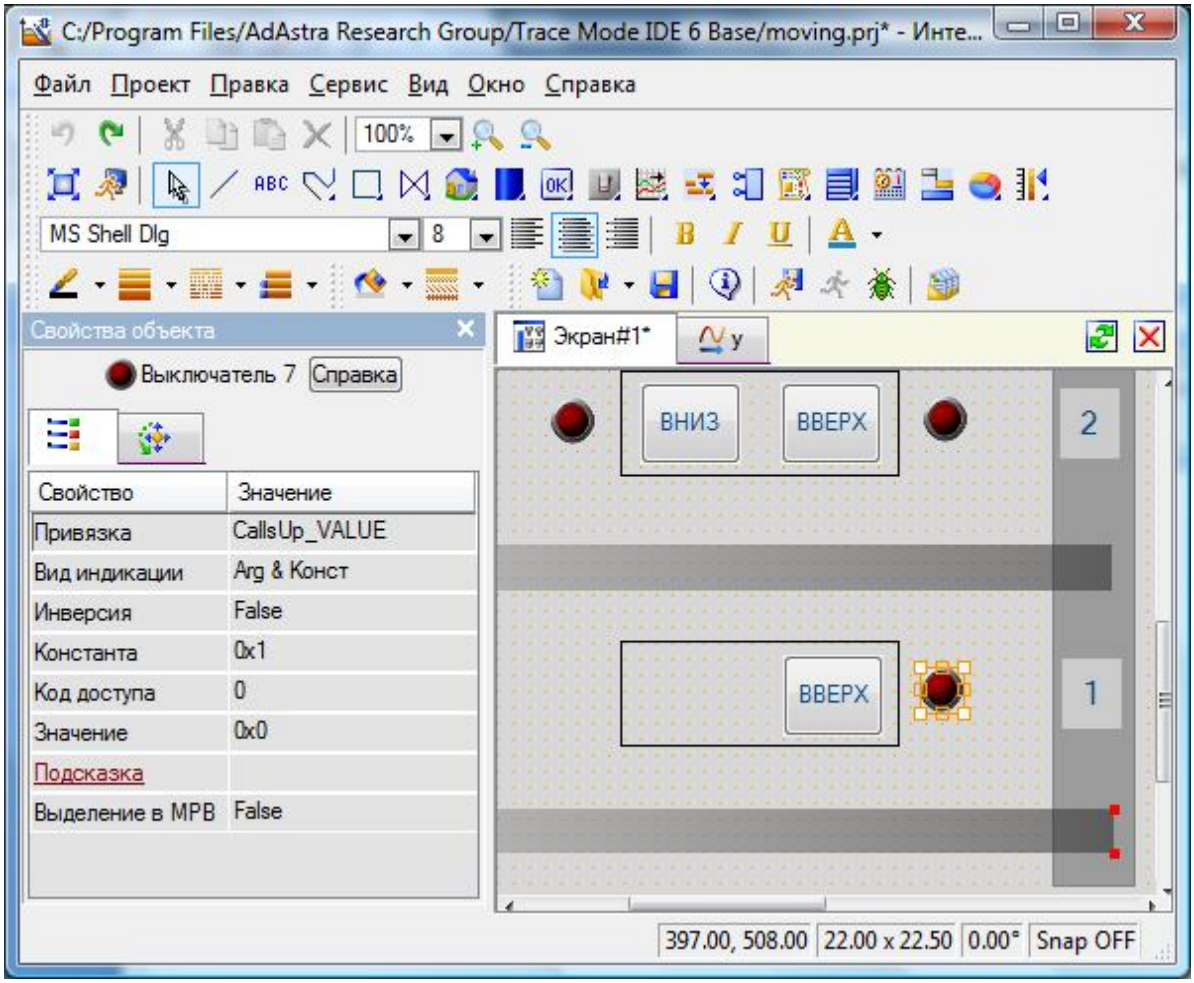


Рис. 60. Настройка индикатора вызова вверх с первого этажа.

Настройка кнопок вызовов и приказов демонстрируется на рис. 61 на примере кнопки вызова вверх с первого этажа. Другие кнопки настраиваются аналогично (табл. 13).

Таблица 13

Настройка кнопок вызовов и приказов

Кнопка	Привязка	Значение
Вызов вверх 1 этаж	CallsUpForm_VALUE	1
Вызов вверх 2 этаж	CallsUpForm_VALUE	2
Вызов вверх 3 этаж	CallsUpForm_VALUE	4
Вызов вниз 2 этаж	CallsDownForm_VALUE	2
Вызов вниз 3 этаж	CallsDownForm_VALUE	4
Вызов вниз 4 этаж	CallsDownForm_VALUE	8

Приказ 1 этаж	OrdersForm_VALUE	1
Приказ 2 этаж	OrdersForm_VALUE	2
Приказ 3 этаж	OrdersForm_VALUE	4
Приказ 4 этаж	OrdersForm_VALUE	8

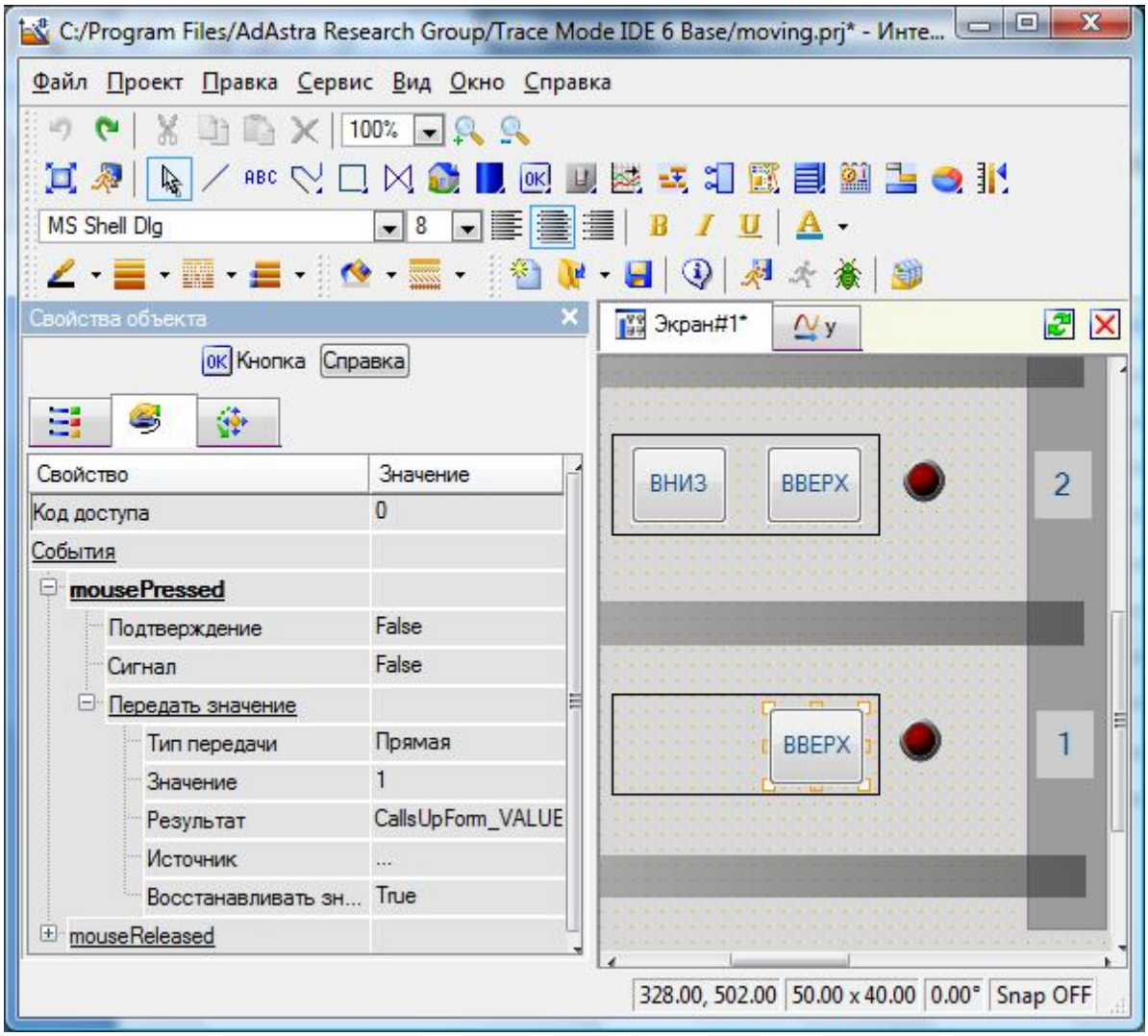


*Рис. 61.*Настройка кнопки вызова вверх с первого этажа.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ASC 300. Руководство пользователя. Преобразователи частоты ACS 300 для регулирования скорости вращения асинхронных двигателей с коротко-замкнутым ротором 0,55…11 кВт. – ABB, 1996.– 62 с.

2. SIEMENS.SIMATIC.Программируемый контроллер S7-200. Системное руководство. – SIEMENS, Издание 06/2004.– 514 с.

3. TRACE MODE v. 6. Справочная система. – AdAstra Research Group, 2008.

4. Руководство пользователя по программированию ПЛК в CoDeSys 2.3. – 3S Smart Software Solution GmbH, Русская редакция: ПК Пролог, 2006. – 453 с.

Андрей Николаевич Рыбалев
доц. кафедры АПП и Э АмГУ, канд. техн. наук, доцент

Программируемые логические контроллеры и аппаратура управления: лабораторный практикум. Часть 4. Системы управления лифтом.

Учебное пособие

Изд-во АмГУ. Подписано к печати ??.??.2015. Формат 60х84/16. Усл. печ. ?,?? Тираж ???. Заказ ???.