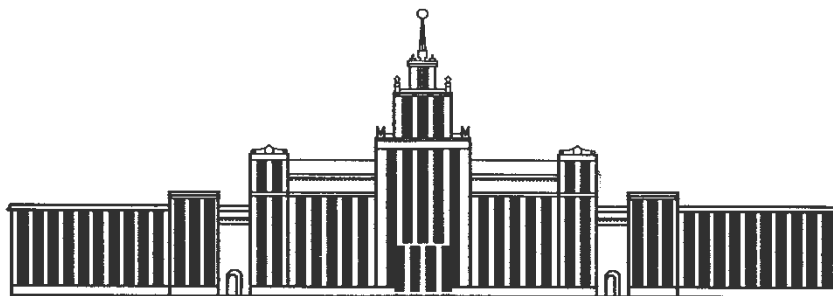

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(национальный исследовательский университет)

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Воронин С.С.

"Программируемые логические контроллеры"

руководство к выполнению практических занятий

**по дисциплине «Микропроцессорные средства в интеллектуальных
мехатронных модулях и робототехнических комплексах»**

Челябинск
2021

Методическое руководство содержит перечень практических работ по программированию логических контроллеров для студентов, обучающихся по направлениям подготовки бакалавриата 15.03.04 «Автоматизация технологических процессов и производств», 15.03.06 «Мехатроника и робототехника» и магистратуры 15.04.04 «Автоматизация технологических процессов и производств», 15.04.06 «Мехатроника и робототехника», 27.04.04 «Управление в технических системах».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
Практическая работа № 1	5
Практическая работа № 2	8
Практическая работа № 3	11
Практическая работа № 4	15
Практическая работа № 5	20
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА	26

ВВЕДЕНИЕ

Данное руководство содержит описание языка программирования LAD для S7-300/400. Первая работа посвящена обзору систем автоматизации S7-300/400 и основам работы со средой TIA Portal. Следующие работы адресованы начинающим пользователям TIA Portal или пользователям, переходящим к TIA Portal от работы с системами управления на базе контакторов и реле.

Описаны базовые функции для дискретного управления с помощью языка программирования LAD и показано, как с помощью двоичных функций преобразуются значения сигналов. Здесь представлены основы двоичных вычислений, работа компаратора, преобразование типов данных. Используя LAD, Вы сможете обрабатывать управляющую программу (управлять ходом выполнения программы) и разрабатывать структурные программы.

Вы сможете создать циклически выполняемую основную программу, Вы также сможете использовать управляемые событиями подпрограммы, такие как подпрограммы, управляемые поведением контроллера при запуске, а также подпрограммы обработки ошибок или проявлений неисправности.

Практическая работа № 1

Программное обеспечение контроллеров SIMATIC. Программы, входящие в пакет TIA PORTAL. Знакомство с TIA PORTAL. Создание проекта. Конфигурирование аппаратной части. Основные настройки

Теоретическая часть

Программируемый контроллер SIMATIC S7-300/400 имеет модульную конструкцию и включает в себя следующие компоненты:

- Стойки (Rack):

стойки используются для размещения в них модулей и для соединения последних друг с другом.

- Источник питания (PS "power supply"):

источник питания обеспечивает внутренние напряжения питания.

- Центральный процессор (CPU "central processing unit"):

центральный процессор используется для размещения и обработки программы пользователя.

- Интерфейсные модули (IM "interface module"):

интерфейсные модули используются для соединения стоек друг с другом.

- Сигнальные модули (SM "signal module"):

сигнальные модули используются для преобразования сигналов, поступающих от процесса, во внутренние сигналы для последующей обработки или в дискретные или аналоговые сигналы для управления приводами.

- Функциональные модули (FM "function module"):

функциональные модули не зависят от CPU, используются для выполнения сложных или зависящих от времени процессов.

- Коммуникационные процессоры (CP "communication processor"):

коммуникационные процессоры используются для связи с подсетями.

- Подсети:

подсети используются для связи программируемых контроллеров друг с другом или с другими устройствами.

Программируемый контроллер (или станция) может состоять из нескольких стоек, которые связываются друг с другом посредством шины. Источник питания, CPU и I/O модули (модули SM, FM и CP) включаются в центральную стойку. Если для I/O модулей недостаточно места или необходимо часть или все I/O модули разместить вне центральной стойки, то в таких случаях используют дополнительные стойки □ стойки расширения, которые соединяются с центральной стойкой посредством интерфейсных модулей. Также возможно подключение к станции распределенных входов/выходов.

Централизованная конфигурация

Программируемый контроллер S7-300 позволяет включить в центральную монтажную стойку до 8 входных/выходных модулей. Если такая однорядная конфигурация контроллера не является достаточной, то возможны два варианта расширения конфигурации при использовании CPU 314 или более мощных процессоров:

- или вариант двухрядной конфигурации, имеющей центральную стойку и одну стойку расширения (при использовании интерфейсных модулей IM 365 и с расстоянием до одного метра между стойками);
- или вариант конфигурации, состоящей максимально из 4 рядов, т.е. кроме центральной стойки, имеющей до 3 стоек расширения (при использовании интерфейсных модулей IM 360 и IM 361 и с расстоянием до десяти метров между стойками).

Вы можете задействовать максимум восемь модулей в стойке. Число модулей может быть ограничено также максимально допустимым током потребления на одну стойку, который составляет 1.2 А (для CPU 312 IFM максимально допустимый ток потребления составляет 0.8 А). Модули связаны между собой внутренней шиной стойки, обеспечивающей функции Р- и К-шин. Для связи модулей друг с другом в стойках служат две шины: шина входов/выходов (I/O или Р-шина) и коммуникационная шина (или К-шина). I/O-шина предназначена для высокоскоростного обмена входными и выходными сигналами, а

коммуникационная шина обеспечивает обмен между модулями большими порциями данных. Коммуникационная шина соединяет CPU и интерфейс программатора (MPI) с функциональными модулями и коммуникационными процессорами.

Локальный сегмент шины

Особую возможность при конфигурировании предоставляет использование прикладного модуля FM 356 из семейства компьютеров для автоматизации M7-300. Модуль FM-356 позволяет "разбить" интерфейсную шину модулей контроллера, чтобы получить контроль над оставшимися в "отсеченном сегменте шины" модулями для автономного управления ими. В данном случае ограничивающим фактором также являются такие параметры, как число модулей и суммарная потребляемая ими мощность.

Практическое задание

Создать новый проект. В пункте «Configure a device» собрать аппаратную конфигурацию, включающую в себя следующие станции:

- 1) Станция S7-300;
- 2) Станция S7-400;
- 3) Две станции удаленного ввода/вывода ET-200S.

Корзину каждой станции необходимо укомплектовать: модулем питания, модулем процессора, модулем входных дискретных сигналов, выходных дискретных сигналов. По желанию добавить модули аналоговых сигналов и коммуникационный модуль.

***ВНИМАНИЕ!** Типы всех модулей выбрать самостоятельно (произвольно) из выпадающего каталога!*

После конфигурирования станций настроить сетевое соединение. Станцию S7-300 и S7-400 соединить по сети Ethernet. Станцию s7-300 и ET-200S, а также станции s7-400 и вторую станцию ET-200S соединить по сети Profibus. Сетевые адреса выбрать произвольно.

Практическая работа № 2

Программирование SIMATIC в среде TIA PORTAL. Языки программирования. Битовые логические инструкции (на примере языков LAD и STL). Знакомство с симулятором контроллера PLCSIM

Теоретическая часть

Битовые логические инструкции работают с двумя числами - 1 и 0. Эти две цифры образуют базис системы счисления, называемой двоичной системой. Цифры 1 и 0 называются двоичными цифрами (binary digits) или просто битами. При работе со схемами, использующими контакты и катушки, значение 1 означает активное состояние или протекание тока, а 0 – неактивное состояние или отсутствие протекания тока. Битовые логические инструкции интерпретируют состояния сигналов 1 и 0 и комбинируют их по правилам булевой логики. Эти комбинации дают результат 1 или 0, называемый Результатом Логической Операции (RLO). Битовые логические инструкции предоставляют в распоряжение следующие функции:

Нормально открытый контакт

- ---| / |--- Нормально замкнутый контакт
- ---(SAVE) Сохранение RLO в бите BR
- XOR ИСКЛЮЧАЮЩЕЕ ИЛИ
- ---() Выходная катушка
- ---(#)--- Промежуточный выход (коннектор)
- ---|NOT|--- Инверсия результата логической операции

Выполнение следующих инструкций производится только при RLO = 1:

- ---(S) Установить выход в 1
- ---(R) Сбросить выход в 0
- SR S/R Триггер
- RS R/S Триггер

Другие инструкции работают при нарастающем или падающем фронте:

---(N)--- Выделение отрицательного фронта RLO

- (P)--- Выделение положительного фронта RLO
- NEG Выделение отрицательного фронта сигнала
- POS Выделение положительного фронта сигнала
- Промежуточное чтение
- Промежуточная запись

---| |--- :(Нормально открытый контакт) будет замыкаться при состоянии бита, указанного в качестве <адреса>, равном 1 . Если состояние сигнала по указанному адресу равно 1, то контакт замкнут, и результат логической операции (RLO) равен 1. Если состояние сигнала по указанному адресу равно 0, то контакт разомкнут, и команда дает результат логической операции (RLO) равный 0.

При использовании команды ---| |--- в последовательной цепи, результат опроса сопрягается с битом RLO по логическому И. При использовании команды - --| |--- в параллельной цепи, результат опроса сопрягается с битом RLO по логическому ИЛИ.

---| / |--- (Нормально замкнутый контакт) будет замыкать цепь при состоянии бита, указанного в качестве <адреса>, равном 0 . Если состояние сигнала по указанному адресу равно 0, то контакт замкнут, и результат логической операции (RLO) равен 1. Если состояние сигнала по указанному адресу равно 1, то контакт разомкнут, и команда дает результат логической операции (RLO) равный 0.

При использовании команды ---| / |--- в последовательной цепи, результат опроса сопрягается с битом RLO по логическому И. При использовании команды - --| / |--- в параллельной цепи, результат опроса сопрягается с битом RLO по логическому ИЛИ.

---[NOT]--- Инструкция инверсия результата логической операции выполняет изменение на противоположное значение результата логической операции RLO.

---() :(Выходная катушка) работает как катушка в цепи управления релейно-контактной схемы. Если к катушке подводится ток ($RLO = 1$), бит <адрес> устанавливается в "1". Если к катушке не подводится ток ($RLO = 0$), бит <адрес>

устанавливается в "0". Выходную катушку можно установить только на правом конце логической цепи. Возможно использование нескольких выходных катушек (максимум 16) . Вы можете инвертировать выход с помощью инструкции---|NOT|--- .

---(R) : (Катушка сброса) Команда сброса выполняется только тогда, когда RLO предыдущей инструкции = 1 (ток поступает на катушку). При протекании тока (RLO = "1"), указанный над катушкой <адрес> сбрасывается в "0". При RLO = "0" (катушка не запитана) инструкция не изменяет статуса указанного операнда. В качестве <адреса> может также использоваться таймер (Т) для сброса его значения в "0" или счетчик (С) для сброса его в "0".

---(S) : (Катушка установки). Инструкция Установить бит выполняется только тогда, когда RLO предыдущей инструкции равен 1. Если RLO равен 1, эта инструкция устанавливает указанный адрес в 1. Если RLO равен 0, то инструкция не влияет на указанный адрес, который остается неизменным.

Практическое задание

Скомпилировать конфигурацию, собранную в задании 1. Проверить отсутствие ошибок при компиляции. Загрузить ТОЛЬКО станцию S7-300 в симулятор.

Программу писать для S7-300.

Вывести в симуляторе окно входного байта IB1 и выходного байта QB1.

В блоке OB1 написать программу (язык LAD или STL), чтобы при замыкании ключей I1.0 – I1.2 зажигались лампочки в соответствии со следующей таблицей

«Замкнуть» ключи			«Зажечь» лампочки
I1.2	I1.1	I1.0	---
0	0	0	Q1.7
0	0	1	Q1.0; Q1.3
0	1	0	Q1.1; Q1.5
0	1	1	Q1.2; Q1.3; Q1.4
1	0	0	Q1.6; Q1.3
1	0	1	Q1.4
1	1	0	Q1.0; Q1.1; Q1.5; Q1.6
1	1	1	Q1.0; Q1.1; Q1.2; Q1.3; Q1.4; Q1.5; Q1.6

Практическая работа № 3

Инструкции сравнения, преобразования, счета, логического управления

Теоретическая часть

Входы IN1 и IN2 сравниваются в соответствии с выбранным Вами типом :

== IN1 равно IN2

<> IN1 не равно IN2

> IN1 больше IN2

< IN1 меньше IN2

>= IN1 больше или равно IN2

<= IN1 меньше или равно IN2

Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Вы можете использовать следующие инструкции сравнения:

- CMP ? I : Сравнение чисел типа Integer
- CMP ? D : Сравнение чисел типа Double Integer
- CMP ? R : Сравнение чисел типа Real

Инструкция CMP ? I : Сравнить целые числа может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Инструкция CMP ? D : Сравнить двойные целые числа может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения,

выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Инструкция CMP ? R: Сравнить числа с плавающей точкой может использоваться как обыкновенный контакт в любом удобном месте контактного плана. Эта инструкция сравнивает входы IN1 и IN2 в соответствии с типом сравнения, выбираемым из окна списка. Если условие сравнения выполняется, то RLO получает значение "1". Он сопрягается с результатами опроса последующих логических операций по схеме И, если они находятся в последовательной цепи и по схеме ИЛИ в случае параллельной цепи.

Практическое задание

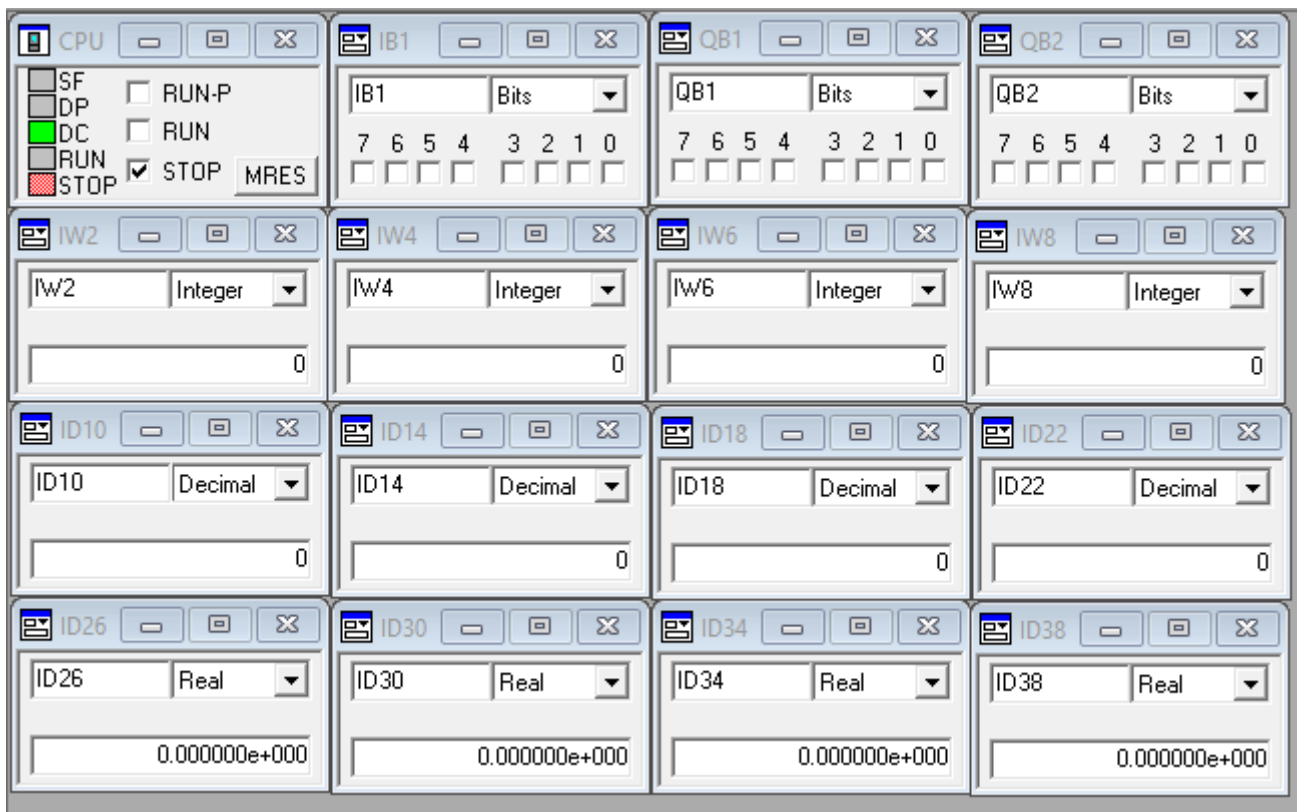
Часть 1

Создать новый проект. В пункте «Configure a device» собрать следующую аппаратную конфигурацию

PS307 5A	CPU 317-2 DP	6ES7 317-2AK14-0AB0	DI16x24VDC Start address = 0	DI64x24VDC Start address = 2	DI64x24VDC Start address = 10	DI64x24VDC Start address = 18	DI64x24VDC Start address = 26	DI64x24VDC Start address = 34	DO 16x24VDC/0.5A Start address = 1
----------	--------------	---------------------	---------------------------------	---------------------------------	----------------------------------	----------------------------------	----------------------------------	----------------------------------	---------------------------------------

Часть 2

Запустить симулятор контроллера PLCSIM. Вывести сигналы следующим образом:



Часть 3

Написать программу для сравнения чисел между собой, реализовать следующую таблицу:

Внимание! Использовать только флаги И1.0, И1.1, И1.2, И1.3 (верхнюю тетраду не трогать)

Флаги (табл. IB1)	Условие	Выходное условие (включить галочки Q)
И1.1 и И1.2	$IW8 > IW4$ Или $IW2 > IW6$	Q2.4, Q1.6, Q1.0
И1.0 или (И1.2 и И1.3)	$ID14 \geq ID10$ И $ID18 = ID22$	Q1.1, Q2.4, Q2.0,
И1.0 и (И1.2 или И1.1)	$IW2 \neq IW6$ И	Q1.0, Q1.7, Q2.7

	ID30>ID26 И ID30<ID34	
И1.1 и И1.3 ИЛИ И1.2 и И1.0	ID22>1000D И ID38<ID26	Q1.6, Q1.3
И1.3 и И1.0 и И1.2	IW4<512D И ID10=ID22 И IW8<>0D	Q2.3, Q2.5, Q1.5
И1.0 и И1.1 и И1.2 и И1.3	{ID10>=ID14 ИЛИ ID10<ID18} И {ID26=ID38 ИЛИ ID34>=ID30}	Q2.2, I2.2

Практическая работа № 4

Математические инструкции. Загрузка и передача данных. Команды управления программой

Теоретическая часть

Используя математические инструкции с целыми числами, Вы можете выполнять операции с двумя числами типа Integer (16 и 32 битовыми):

- ADD_I : Сложение целых чисел
- SUB_I : Вычитание целых чисел
- MUL_I : Умножение целых чисел
- DIV_I : Деление целых чисел
- ADD_DI : Сложение двойных целых чисел
- SUB_DI : Вычитание двойных целых чисел
- MUL_DI : Умножение двойных целых чисел
- DIV_DI : Деление двойных целых чисел
- MOD_DI : Получение остатка от деления двойных целых чисел

ADD_I :инструкция Сложить целые числа активируется при состоянии сигнала 1 на входе EN (деблокировка входа) . Эта инструкция складывает входы IN1 и IN2 и результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1 , а выход ENO равен 0 и таким образом, каскадное включение следующих инструкций не будет выполняться. SUB_I: инструкция Вычитание целых чисел активируется при состоянии сигнала 1 на входе EN (деблокировка входа). Эта инструкция вычитает из значения на входе IN1 значение IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

MUL_I : инструкция Умножение целых чисел активируется при состоянии сигнала 1 на входе EN (деблокировка входа). Эта инструкция умножает значения поданные на входы IN1 и IN2, а результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

DIV_I : инструкцию Деление целых чисел активирует состояние сигнала 1 на входе EN (деблокировка входа). Эта инструкция делит значение, поданное на вход IN1 на IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны

1, а значение ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

ADD_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию Сложить двойные целые числа. Эта инструкция складывает входы IN1 и IN2. Результат можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

SUB_DI: Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию Вычитание двойных целых чисел. Эта инструкция вычитает значения поданные на входы IN1 и IN2. Результат можно считать на выходе OUT. Если результат выходит за пределы допустимого диапазона для целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

MUL_DI : инструкция Умножение двойных целых чисел активируется при состоянии сигнала 1 на входе EN (деблокировка входа). Эта инструкция перемножает входы IN1 и IN2. Результат в виде 32-битного целого числа можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойных целых чисел, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

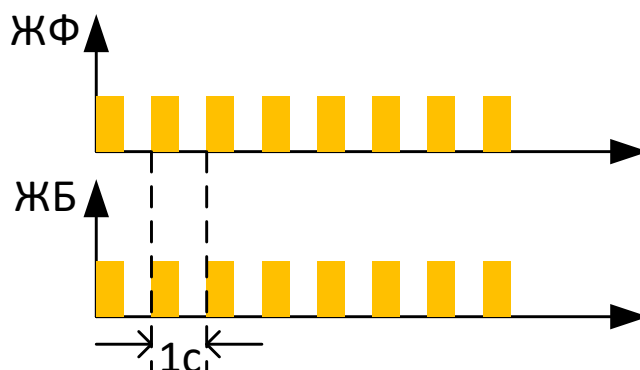
DIV_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию Разделить двойные целые числа. Эта инструкция делит вход IN1 на IN2. Частное от деления (округленный результат) можно считать на OUT. Инструкция Разделить двойные целые числа хранит частное от деления в виде единственного 32-битного значения в формате DINT. Эта инструкция не выдает остатка от деления. Если частное выходит за пределы допустимого диапазона для двойного целого числа, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

MOD_DI :Состояние сигнала 1 на входе EN (деблокировка входа) активирует инструкцию получить остаток от деления двойных целых чисел. Эта инструкция делит вход IN1 на IN2. Остаток от деления можно считать на OUT. Если результат выходит за пределы допустимого диапазона для двойного целого числа, то биты OV и OS слова состояния равны 1, а ENO равно 0 и ,таким образом, каскадное включение следующих инструкций не будет выполняться. Смотрите также оценку бит слова состояния при выполнении математических инструкций.

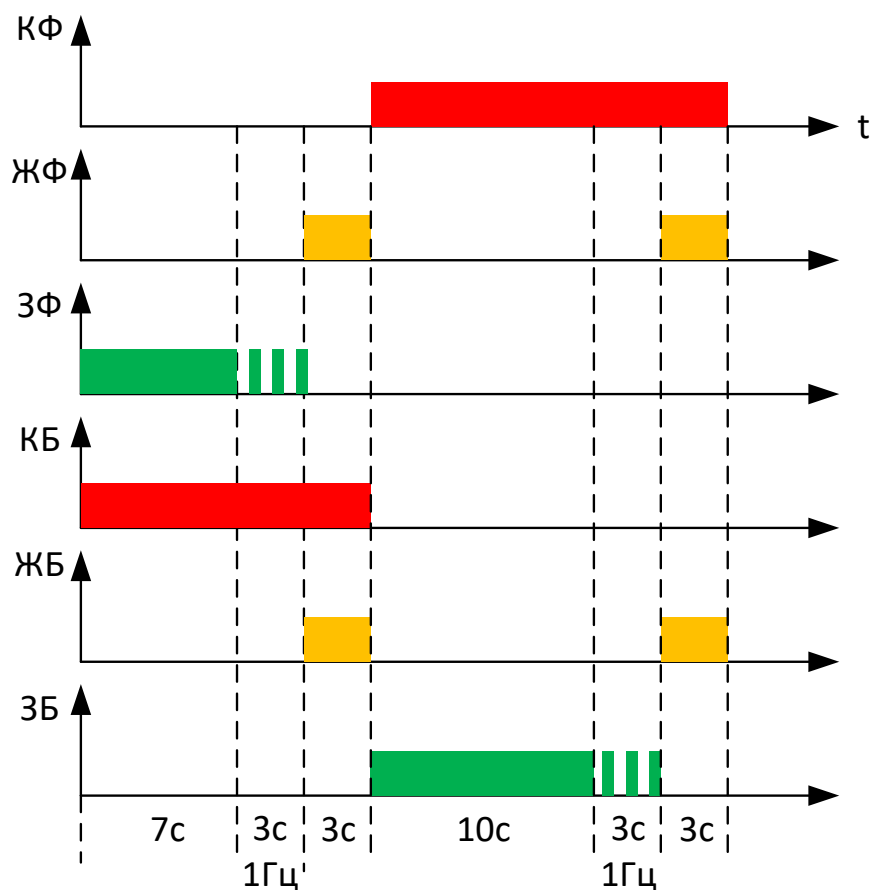
Практическое задание

Необходимо запрограммировать светофор для следующих режимов работы: режим «мигающий желтый сигнал», режим «Работа» (светофор + цифровой таймер).

1) При включении режима «мигающий желтый сигнал» (Flashing Yellow), мигает желтый сигнал (все стороны) с частотой 1 Гц, как показано на диаграмме ниже



2) При включении режима «Работа» (Working Mode 1):



Когда должны включаться сигналы цвета секундомера и пешехода – определить самостоятельно.

Реализация светофора: при помощи функций контроллера, описанного в практических занятиях 1-4

Таблица сигналов:

Входные сигналы	
I0.0	Режим работы "Flashing Yellow"
I0.1	Режим работы "Working Mode 1"
I0.2	Режим работы "Working Mode 2"
Выходные сигналы	
Q0.0	Включить зеленый фронтальный
Q0.1	Включить желтый фронтальный
Q0.2	Включить красный фронтальный
Q0.3	Включить зеленый боковой
Q0.4	Включить желтый боковой
Q0.5	Включить красный боковой
Q0.6	Секундомер красного цвета
Q0.7	Секундомер зеленого цвета
Q1.0	Пешеход "зеленый"
Q1.1	Пешеход "красный"
QW2	Показания секундомера

Практическая работа № 5

Режимы работы контроллера SIMATIC. Способы переключения режимов работы.

Приоритет режимов работы

Теоретическая часть

1. Режим начальной установки

Режим начальной установки используется при включении микропроцессора (МП). В режиме начальной установки счетчик команд РС микропроцессора обнуляется, а на шине адреса ША, следовательно, выставляется шестнадцатеричный адрес ячейки памяти 0000. Дальнейшие действия МП, как известно, сводятся к реализации цикла М1 (чтение кода операции). Следовательно, в «нулевой» ячейке памяти должна быть записана любая команда, с которой начинается выполнение программы. Чаще всего в эту ячейку записывают команду безусловного перехода к области памяти, где располагается управляющая программа системы. Таким образом, режим начальной установки обеспечивает запуск МП.

Практически режим начальной установки реализуется подачей на вход Rst (Reset) МП сигнала низкого уровня, который может быть сформирован либо в момент включения питания, либо нажатием кнопки сброс, подключенной ко входу Rst.

2. Режим ожидания

Большинство современных микропроцессоров и микроконтроллеров имеют специальные режимы работы, предназначенные для снижения энергопотребления. Самым распространенным из них является режим ожидания (idle mode). В этом режиме часть ядра микропроцессора, выполняющая команды, останавливается, а все периферийные устройства и прерывания продолжают работать.

В ряде случаев быстродействие МП оказывается несогласованным с быстродействием внешнего устройства (ВУ). Например, быстродействие интегральных схем (ИС) памяти или устройства цифropечати существенно ниже

быстродействия МП. В этих случаях необходимо приостановить действие МП, т. е. растянуть машинный цикл на целое количество тактов.

Аналогичная ситуация может возникнуть при реализации поциклового и покомандного режимов работы МП. Необходимость работы МП в этих режимах возникает при отладке программ, когда после выполнения каждого машинного цикла или после выполнения команды требуется переводить МП в режим ожидания.

Переключение в режим ожидания и выход из него требует небольших временных затрат и может происходить несколько раз в миллисекунду. Каждый раз, когда операционная система обнаруживает, что все процессы заблокированы в ожидании прерывания, события или тайм-аута, она должна перевести микропроцессор в режим ожидания, чтобы уменьшить энергопотребление. Так как любое прерывание может вывести микропроцессор из режима ожидания, использование этого режима позволяет программному обеспечению осуществлять интеллектуальное ожидание событий в системе. Однако не все так просто и для максимальной производительности этот способ требует тщательной разработки программы.

Допустим, мы написали код, который опрашивает регистр состояния и ожидает установки флага. Возможно, мы проверяем флаг статуса FIFO буфера в последовательном порту, чтобы узнать, были ли получены данные. Или, возможно, мы контролируем двухпортовую ячейку памяти, чтобы узнать, записал ли другой процессор или устройство в системе переменную, предоставляя нам контроль над совместно используемым ресурсом. При всей полезности данного процесса, циклический опрос регистра лишает нас возможности продлить срок службы батареи/аккумулятора.

Если событие не может быть непосредственно привязано к внешнему прерыванию, переключение микропроцессора в режим ожидания и выход из него можно выполнять по сигналу системного таймера. Этот вариант все еще предпочтительней циклического опроса. Например, вы ожидаете какое-то событие

и знаете что обработаете его, проверяя статус события каждую миллисекунду. Включите системный таймер на 1 миллисекунду и переведите микропроцессор в режим ожидания. Каждый раз, когда запускается прерывание проверяйте статус события. Если статус не изменился, переводите микропроцессор снова в режим ожидания.

Подавляющее большинство современных карманных компьютеров и смартфонов работает на микропроцессорах и операционных системах, использующих режим ожидания. Фактически, большинство из этих устройств переключается в режим ожидания и выходит из него многократно в течение секунды; они выходят из этого режима всякий раз, когда кто-то дотрагивается до сенсорного экрана, нажимает клавишу или когда возникает какое-либо событие.

3. Режим прерывания программы

Используемый для обмена данными с ВУ режим ожидания имеет очевидный недостаток - снижение производительности МП за счет бесполезной траты времени на ожидание готовности ВУ. Режим прерывания программы не имеет этого недостатка и является особенно полезным для организации работы МС в реальном масштабе времени. Сущность режима прерывания заключается в том, что ВУ обращается с запросом на обмен данными тогда, когда оно готово обменяться данными. Инициатором обмена данными является ВУ, которое выдает сигнал запроса прерывания асинхронно по отношению к МП в виде сигнала ЗПР высокого уровня. Запрос на прерывание может быть воспринят МП не всегда и может быть удовлетворен только в том случае, если до поступления сигнала ЗПР триггер запроса прерывания установлен. Для установки триггера используется специальная команда EI (разрешение прерывания). Если же после режима начальной установки команда EI не была выполнена, то запросы ВУ на прерывания игнорируются.

4. Режим прямого доступа к памяти

Многие МС имеют в своем составе ВУ с высокой скоростью передачи больших массивов информации (например, накопители на гибких магнитных дисках). В этом случае обмен данными с ВУ организуется в режиме прямого

доступа к памяти (ПДП). Суть режима заключается в том, чтобы осуществить обмен информацией между ВУ и памятью МС, минуя микропроцессор. Для организации режима ПДП в МП предусмотрен диалоговый обмен управляющими сигналами между ВУ и МП.

Когда ВУ инициирует запрос на ПДП, микропроцессор приостанавливает выполнение основной программы и переводит буферы шин адреса и данных в такое состояние, которое означает отключение шин ША и ШД от микропроцессора. Шинами начинает управлять контроллер ПДП, организуя обмен данными между ВУ и памятью МС.

5. Режим останова

Как указывалось, МП работает в циклическом режиме, когда после выполнения команды МП инициирует цикл М1 — чтение кода операции очередной команды. Остановить естественный процесс функционирования возможно специальной командой останова.

Выполнение программы прекращается и в состоянии останова микропроцессор может находиться как угодно долго. Из состояния останова МП выводится следующими способами:

путем подачи сигнала высокого уровня на вход сброса СБР с продолжительностью не менее трех периодов синхронизации. Когда на линии СБР после этого устанавливается низкий уровень, то по нарастающему фронту сигнала Ф1 генерируется внутренний сигнал сброса. Он загружает в счетчик РС нули и заставляет устройство управления сформировать следующий такт Т1 машинного цикла М1 выборки кода операции. Следовательно, МП обращается к ячейке памяти, которая обычно является начальным адресом подпрограммы;

путем подачи сигнала высокого уровня на вход прерывания ЗПР.

Особенностью режима останова является возможность входа в режим ПДП по наличию высокого уровня сигнала ЗХ. Запрос на ПДП не будет удовлетворяться только в том случае, если был уже подан сигнал ЗПР (запрос на прерывание), но не было еще подтверждения прерывания сигналом РПР. После подтверждения

прерывания возможен вход в режим ПДП. Таким образом, микропроцессор может работать в различных режимах информационного обмена. Выбор режима определяется быстродействием внешнего устройства и объемом передаваемых данных. Для поддержки каждого из режимов в составе микропроцессорного комплекта предусмотрены специальные интерфейсные микросхемы.

Практическое задание

Конфигурацию выбрать произвольно

В качестве исходных данных использовать двойное слово MD10

Реализовать условия:

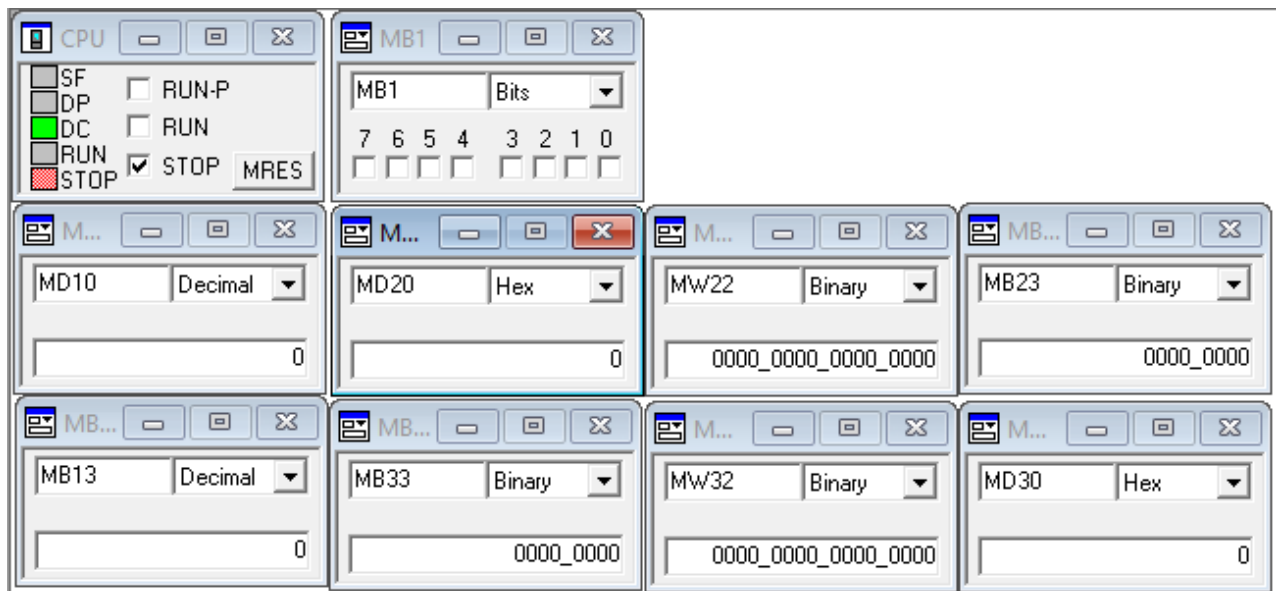
- 1) Если включить M1.0, то осуществляется передача MD10 в двойное слово MD20;
- 2) Если включить M1.1, то осуществляется передача MD10 в слово MW22;
- 3) Если включить M1.2, то осуществляется передача MD10 в байт MB23.

Далее, старший байт двойного слова (MB13) использовать для передачи:

- 1) Если включить M1.4, то осуществляется передача MB13 в байт MB33;
- 2) Если включить M1.5, то осуществляется передача MB13 в слово MW32;
- 3) Если включить M1.6, то осуществляется передача MB13 в двойное слово MD30.

ОДНОВРЕМЕННО ДОЛЖНО ВЫПОЛНЯТЬСЯ ТОЛЬКО ОДНО ПРЕОБРАЗОВАНИЕ (только одна галочка)

В симуляторе PLCSIM вывести:



РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Кангин, В. В. Промышленные контроллеры в системах автоматизации технологических процессов Текст учеб. пособие для вузов по направлению "Автоматизация технол. процессов и пр-в" В. В. Кангин. - Старый Оскол: Тонкие наукоемкие технологии, 2013. - 407 с. ил.

2. Медведев, М. Ю. Программирование промышленных контроллеров Текст учеб. пособие для магистров техники и технологии вузов по направлению "Электротехника, электромеханика и электротехнологии" М. Ю. Медведев, В. Х. Пшихопов. - СПб. и др.: Лань, 2011. - 288 с. ил.

3. Шишов, О. В. Программируемые контроллеры в системах промышленной автоматизации Текст ## учебник... О. В. Шишов. - #. - М.: ИНФРА-М, 2017. - 363, [2] с. ил.

4. Бродин, В. Б. Микропроцессор i 486. Архитектура, программирование, интерфейс. - М.: ДИАЛОГ-МИФИ, 1993. - 238,[2] с. ил.

5. Гилмор, Ч. Введение в микропроцессорную технику Пер. с англ. В. М. Кисельникова и др. - М.: Мир, 1984. - 334 с. ил.

6. Левенталь, Л. Введение в микропроцессоры: Программное обеспечение, аппаратные средства, программирование Пер. с англ. под ред. В. В. Сташина. - М.: Энергоатомиздат, 1983. - 464 с. ил.

7. Гуров, В. В. Архитектура микропроцессоров Текст учеб. пособие для вузов В. В. Гуров. - М.: Интернет-ун-т информационных технологий : Бином. Лабор, 2010

8. Паппас, К. Микропроцессор 80386 Справочник Перевод с англ. И. П. Пчелинцева, С. В. Комягина; Под ред. В. В. Василькова. - М.: Радио и связь, 1993. - 318 с. ил.