Старооскольский технологический институт
(филиал) Федерального государственного образовательного
учреждения высшего профессионального образования
«Государственный технологический университет
«Московский институт стали и сплавов»

Кафедра АиПЭ

Д.А. Полещенко, Кривоносов В.А.

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ И ПРОИЗВОДСТВ

методические указания
к выполнению лабораторных работ
для студентов специальностей
220301 – «Автоматизация технологических процессов и производств»,
230201 – «Информационные системы и технологии»,
140604 – «Электропривод и автоматика промышленных установок и технологических комплексов»,
230102 – «Автоматизированные системы обработки информации и управления»
(очная форма обучения)

Одобрено редакционно-издательским советом

Старый Оскол
2008

УДК 519.7
ББК 6*3,1

Рецензент: декан ФАИТ, зав. каф. АиПЭ, профессор, д.т.н. Еременко Ю.И.

Полещенко Д.А., Кривоносов В.А. Автоматизация технологических процессов и производств. Методические указания к выполнению лабораторных работ. Старый Оскол, СТИ МИСиС, 2008. – 60 с.

Методические указания к выполнению лабораторных работ по курсу «Автоматизация технологических процессов и производств» для студентов специальностей: 220301 – «Автоматизация технологических процессов и производств», 230201 – «Информационные системы и технологии», 140604 – «Электропривод и автоматика промышленных установок и технологических комплексов», 230102 – «Автоматизированные системы обработки информации и управления», очной формы обучения.

Содержание

Предисловие

Автоматизация технологических процессов и производств является одним из основных направлений повышения эффективности, экологической безопасности, безаварийности производств. Автоматизированные системы управления технологическими процессами (АСУ ТП) обеспечивают снижение удельных затрат сырья и энергии, повышение ресурса технологического оборудования за счет оптимизации производственных режимов, предупреждения аварийных ситуаций, рациональной загрузки агрегатов.

Современные АСУ ТП являются распределенными, иерархическими системами управления. Основой таких АСУ ТП служат микропроцессорные устройства, осуществляющие сбор и обработку информации на нижних уровнях иерархии управления, а также рабочие станции с установленными на них SCADA системами на верхних уровнях.

Лабораторный практикум по дисциплине «Автоматизация технологических процессов» включает цикл работ по построению системы автоматизированного управления температурным режимом в электрической нагревательной печи на основе промышленных микропроцессорных контроллеров.

В процессе выполнения лабораторной работы №1 студенты осваивают все базовые процедуры и операции по подготовке к работе и программированию контроллера Simatic S7-300. В лабораторной работе №2 производится ознакомление с интеграцией контроллера Simatic S7-300 со SCADA системой PROTOOL. В работе №3 студенты обучаются программировать базовые инструкции STEP 7 и знакомятся с функционированием таймеров на примере реализации широтно-импульсной модуляции аналогового сигнала. В лабораторной работе №4 производится экспериментальное исследование и построение математической модели объекта управления. В работе №5 определяются оптимальные параметры ПИ-регулятора. В работе №6 осуществляется построение и экспериментальная проверка качества функционирования системы управления температурным режимом на основе контроллера Simatic S7-300.

По каждой из лабораторных работ студенты оформляют индивидуальные отчеты. В процессе защиты отчета студент должен ответить на вопросы и выполнить практическое задание.

Лабораторная работа №1
Ввод аналоговых сигналов в Simatic S7-300

1.1 Цель работы: получить навык ввода и калибровки аналогового сигнала в контроллер Simatic S7-300.

1.2 Теоретическое введение

Программируемый контроллер SIMATIC S7-300

Программируемый контроллер S7-300 имеет модульное построение. Модули, из которых он состоит, могут быть центральными (расположенными рядом с CPU) или распределенными (расположенными рядом с объектом управления) без обязательных специальных установок или назначений параметров.

S7-300 включает в себя следующие компоненты:

- *Стойки* (Racks) – служат для размещения модулей и соединения их между собой;
- *Электропитание* (Power Supply, PS) – обеспечивает подачу электроэнергии к внутренним устройствам;
- *Центральное процессорное устройство* (Central Processing Unit, CPU) – хранит и обрабатывает программу пользователя;
- *Интерфейсные модули* (Interface Modules, IM) – соединяют стойку с другой стойкой;
- *Сигнальные модули* (Signal Modules, SM) – адаптируют системные сигналы к внутреннему уровню сигнала или управляют приводами посредством цифровых или аналоговых сигналов;
- *Функциональные модули* (Function Modules, FM) – выполняют сложную или критичную по времени обработку независимо от CPU;
- *Коммуникационные процессоры* (Communications Processors, CP) – устанавливают соединения со вспомогательными сетями (подсетями);
- *Подсети* (Subnets) – соединяют программируемые контроллеры друг с другом и с другими устройствами.

Области памяти CPU

В системах SIMATIC S7 распределенный вход/выход (I/O) является интегрированной частью системы. CPU с его различными областями памяти формирует аппаратную основу для обработки пользовательских программ.

Пользовательская память

На рис. 1 показаны области памяти CPU, важные для программы. Сама пользовательская программа находится в двух областях, которые называются загрузочная память и рабочая память.

Загрузочная память может быть интегрированной в CPU или подключаемой (plugin) картой памяти. Вся программа пользователя, включая конфигурационные данные, располагается в загрузочной памяти.

Рабочая память представляет собой высокоскоростную RAM полностью встроенную в CPU. Рабочая память содержит связанные части

пользовательской программы; в основном это программный код и пользовательские данные.

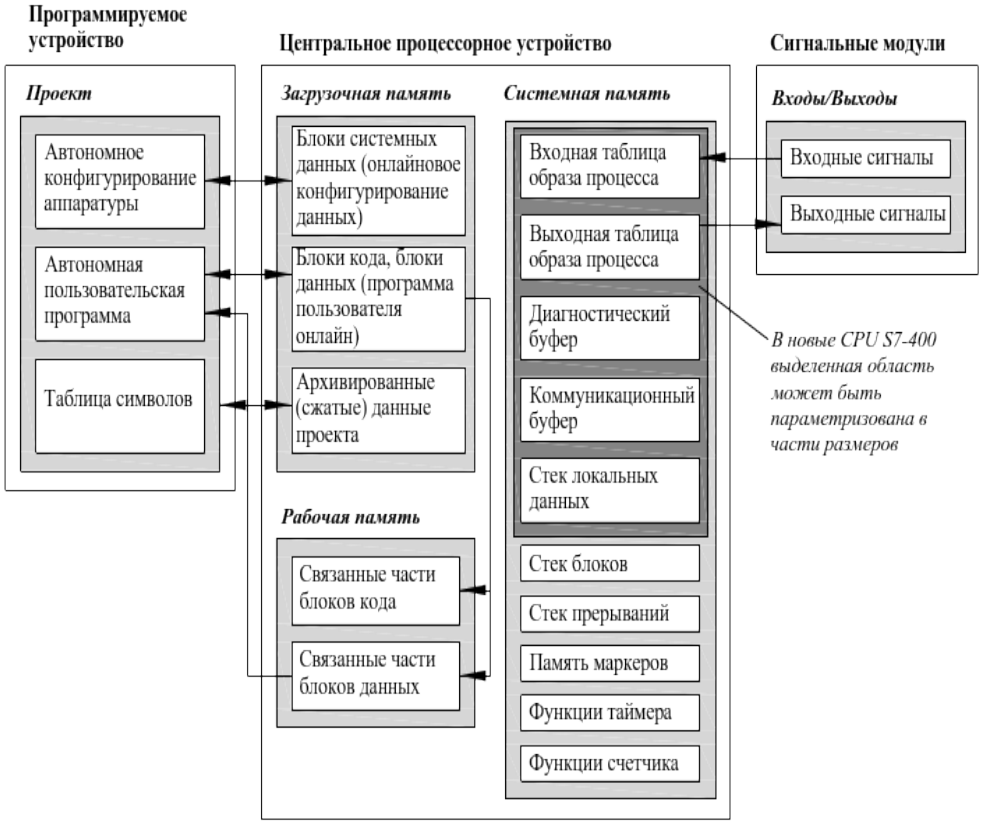


***Рис. 1.1** Области памяти CPU*

Программирующее устройство (программатор) переносит всю программу, включая данные конфигурации, в загрузочную память. Затем операционная система CPU копирует программный код и пользовательские данные в рабочую память. При обратной загрузке программы в программирующее устройство выбранные из загрузочной памяти блоки дополняются текущими значениями адресов данных из рабочей памяти.

Если загрузочная память состоит из RAM, для хранения программы пользователя требуется резервная батарея. Там, где загрузочная память реализована как интегрированная EEPROM или подключаемая карта флэш-памяти EPROM, CPU может работать без резервной батареи.

В состав загрузочной памяти CPU 3xxIFM входят RAM- и EEPROM-компоненты. Программа перемещается и тестируется в памяти RAM, а затем посредством команды меню протестированная программа может храниться в интегральной (встроенной) памяти EEPROM, защищенной от сбоев питания.

CPU S7-300 (за исключением CPU 318) снабжены встроенной загрузочной памятью RAM, способной хранить всю программу. Карту флэш-памяти EPROM можно использовать в качестве носителя данных или защищенной от сбоев электропитания среды хранения для программы пользователя.

В S7-300 текущие значения частей пользовательской памяти (блоки данных – data blocks) и системной памяти (память меркеров, таймеры и счетчики) могут храниться на энергонезависимом элементе памяти. Таким образом, можно сохранить и обезопасить данные в случае сбоя питания без резервного источника питания.

Системная память

Системная память (system memory) содержит адреса (переменные), к которым обращаются в программе. Адреса объединены в области (области адресов), которые содержат определяемое центральным процессорным устройством количество адресов. Адреса могут быть, к примеру, входами, используемыми для сканирования сигнального состояния переключателей мгновенного контакта (кнопок) и переключателей ограничения (конечных выключателей), и выходами, которые можно задействовать для управления контакторами и лампами.

Системная память CPU содержит следующие области адресов:

- Входы (Inputs, I) – являются образом («образом процесса») модулей цифрового входа.
- Выходы (Outputs, Q) – являются образом («образом процесса») модулей цифрового выхода.
- Память меркеров (Bit memory, M) – хранит информацию, доступную всей программе.
- Таймеры (Timers, T) – это ячейки памяти, используемые для реализации интервалов ожидания и мониторинга.
- Счетчики (Counters, C) – это организуемые на программном уровне ячейки памяти, используемые для ведения счета по возрастанию и убыванию.
- Временные локальные данные (Temporary local data, L). Ячейки памяти, используемые в качестве динамических промежуточных буферов во время обработки блоков. Временные локальные данные расположены в L-стеке, которые динамически используются CPU во время выполнения программы.

Различают бинарные (дискретные) и аналоговые сигналы.

Бинарные (дискретные) сигналы

Бинарный сигнал (binary signal) содержит один бит информации. Примерами дискретных сигналов являются входные сигналы от конечных выключателей, переключателей мгновенного контакта и т.п., которые поступают на цифровые входные модули, и выходные сигналы, которые управляют лампами, контакторами и т.п. через цифровые выходные модули.

Аналоговые сигналы

Аналоговый сигнал (analog signal) содержит 16 бит информации. Аналоговый сигнал соответствует «каналу», который отображается в контроллере в виде машинного слова (word), то есть двух байт. Аналоговые входные сигналы (например, напряжения от терморезисторов) поступают в аналоговые входные модули, оцифровываются и после этого становятся доступными для обработки в контроллере в виде 16-разрядного сигнала (16

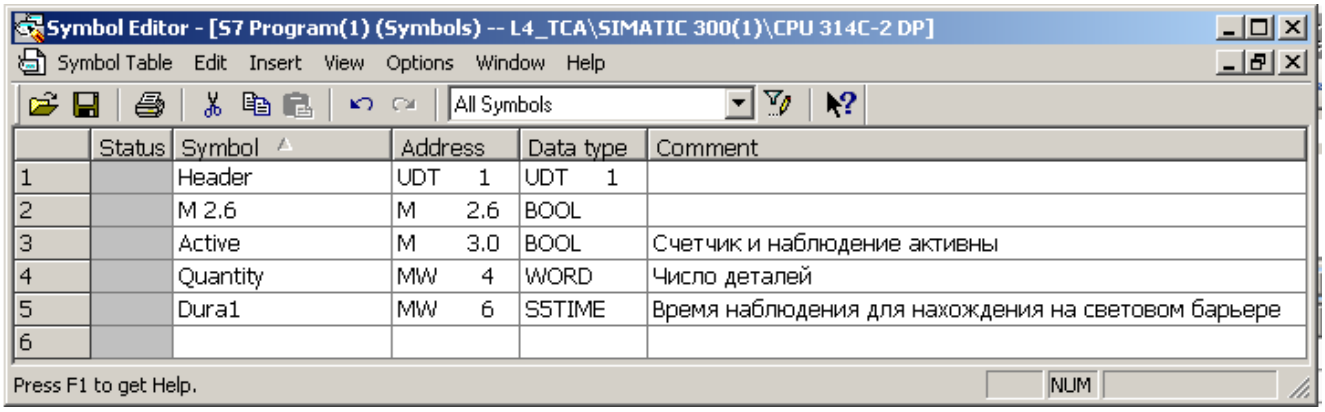
информационных битов). С другой стороны, 16-разрядный сигнал может управлять аналоговым индикатором посредством аналогового выходного модуля, где информация преобразовывается в аналоговую величину (например, ток).

Таблица символов

В управляющей программе работают с адресами; это входы, выходы, таймеры, блоки. Можно присвоить абсолютные адреса (например, I1.0) или символические адреса (например, Start signal). Символическая адресация использует вместо абсолютного адреса имена. Применяя осмысленные имена, можно сделать программу более читаемой.

В символической адресации различают *локальные* (*local*) и *глобальные* (*global*) символы. Локальный символ известен только в блоке, в котором он был определен. Можно использовать одинаковые локальные символы в разных блоках для различных целей. Глобальный символ известен во всей программе и имеет одинаковое значение во всех блоках. Глобальные символы определяются в таблице символов (объект *Symbols* (*Символы*) в контейнере *S7 Program* (*S7-программа*)).

На рис. 1.2 представлен пример таблицы символов.



	Status	Symbol	Address	Data type	Comment
1		Header	UDT 1	UDT 1	
2		M 2.6	M 2.6	BOOL	
3		Active	M 3.0	BOOL	Счетчик и наблюдение активны
4		Quantity	MW 4	WORD	Число деталей
5		Dura1	MW 6	S5TIME	Время наблюдения для нахождения на световом барьере
6					

***Рис. 1.2** Пример таблицы символов*

В таблице символов можно присвоить имена следующим адресам и объектам:

- Входам I, выходам Q, периферийным входам PI и периферийным выходам PQ;
- Меркерам M, таймерам T и счетчикам C;
- Кодовым блокам OB, FB, FC, SFC, SFB и блокам данных DB;
- Определенным пользователем типам данных UDT;
- Таблице переменных (variable table) VAT.

Адреса данных и блоки данных включены в число локальных адресов; ассоциированные символы определяются в разделе описаний блока данных в случае глобальных блоков данных и разделе описаний функционального блока в случае блоков данных экземпляров.

Редактор программ

Для создания пользовательских программ базовый пакет STEP 7 содержит редактор программ (Program Editor) для языков программирования LAD (*ladder logic* или *ladder diagram* – контактный план; представление, схожее с диаграммами релейной логики; многоступенчатая схема), FBD (*function block diagram* – диаграмма функциональных блоков или функциональный план) и STL (*statement list* – список операторов или список мнемоник; ассемблероподобный язык). Программирование на языках LAD и FBD осуществляется путем ввода блоков из существующей библиотеки и установки на их входах и выходах соответствующих адресов.

Если используется символическая адресация для глобальных адресов, то символы уже должны быть присвоены абсолютным адресам; тем не менее, можно ввести новые символы или изменить их во время ввода программы.

Блоки

С целью повышения удобочитаемости и понимания программы можно разбить ее на произвольное число разделов. Языки программирования STEP 7 поддерживают эту концепцию и предоставляют необходимые функции. Каждая часть программы должна быть независимой и обладать технологическим или функциональным базисом. Эти разделы программы называются «блоками» («Blocks»). Блок – это раздел программы, который определяется собственной функциональностью, структурой или решаемой задачей.

Типы блоков

Step7 предоставляет для разных задач различные типы блоков:

- *Пользовательские блоки* (user blocks) – содержат пользовательскую программу и пользовательские данные.
- *Системные блоки* (system blocks) – содержат системную программу и системные данные.
- *Стандартные блоки* (standard blocks) – готовые к непосредственному использованию (созданные заранее) блоки, такие как драйверы для функциональных модулей (FM) и коммуникационных процессоров (CP).

Пользовательские блоки

В случае больших и сложных программ рекомендуется и отчасти является необходимостью «структурирование» (разбиение) программы с выделением блоков. В зависимости от приложения можно выбрать для использования различные типы блоков:

Организационные блоки (*Organization blocks – OB*) – служат в качестве интерфейса между операционной системой и программой пользователя. Операционная система CPU вызывает организационные блоки при возникновении определенных событий, например, в случае аппаратного прерывания или прерывания по времени суток. Главная программа находится в организационном блоке OB1. Остальные организационные блоки имеют постоянные номера, назначенные в зависимости от событий, для обработки которых они вызываются.

Функциональные блоки (*Function blocks – FB*) – являются частями программы, вызовы которых могут быть запрограммированы с помощью параметров блока. У них есть область памяти для переменных (variable memory), которая расположена в блоке данных. Этот блок постоянно назначен функциональному блоку или, точнее, *вызову* (*call*) функционального блока.

Кроме того, каждому вызову функционального блока можно назначить другой блок данных (с такой же структурой данных, но содержащий другие значения). Подобный постоянно назначенный блок называется экземплярным блоком данных или экземпляром блока данных (instance data block), а совокупность вызова функционального блока и экземплярного блока данных называется экземпляром вызова (call instance) или просто «экземпляром» («instance»). Функциональные блоки могут также хранить свои переменные в экземплярном блоке данных вызывающего функционального блока; такой экземпляр называется «локальным экземпляром» («local instance»).

Функции (*Functions – FC*) – используются для программирования часто повторяющихся или сложных функций автоматизации. Для них могут быть назначены параметры. Функции могут возвращать значение (называемое значением функции) в вызывающий блок. Значение функции является необязательным параметром. Кроме него у функций могут быть другие выходные параметры. Функции не сохраняют информацию и не имеют назначенного блока данных.

Блоки данных (*Data blocks – DB*) – содержат данные программы. Программируя блоки данных, определяют форму хранения данных (в каком блоке, в каком порядке и какой при этом используется тип данных). Блоки данных используются двумя способами:

1) в качестве глобальных блоков данных (global data blocks),
2) в качестве экземплярных блоков данных (instance data blocks).

Глобальный блок данных в пользовательской программе является, так сказать, «свободным» блоком данных и не назначается кодовому блоку. Однако экземплярный блок данных назначен функциональному блоку и хранит часть локальных данных этого блока.

Количество блоков определенного блочного типа и размер блоков зависит от типа CPU. Число организационных блоков и их номера фиксированы; они назначаются операционной системой CPU. Блокам других типов можно самостоятельно назначить номера из определенного диапазона. Также можно с помощью таблицы символов назначить каждому блоку имя (символ) и затем обращаться к блокам по присвоенному имени.

Системные блоки

Системные блоки являются компонентами операционной системы. Они могут содержать программы (системные функции, SFC, или системные функциональные блоки, SFB) или данные (системные блоки данных, SDB). Системные блоки предоставляют доступ к важным системным функциям,

таким как управление внутренними таймерами CPU или различные коммуникационные функции.

Можно вызвать SFC и SFB, но нельзя ни изменить их, ни самостоятельно запрограммировать. Сами блоки не занимают места в пользовательской памяти (user memory); только вызовы блоков и экземплярные блоки данных блоков SFB располагаются в пользовательской памяти.

Блоки SDB содержат информацию о таких вещах, как конфигурация системы автоматизации или параметры модулей. Система STEP 7 сама генерирует эти блоки и управляет ими. Тем не менее, можно определять их содержимое, например, при конфигурировании станций. Как правило, блоки SDB располагаются в загрузочной памяти (load memory). Из пользовательской программы доступ к ним получить нельзя.

Стандартные блоки

В дополнение к создаваемым функциям и функциональным блокам можно использовать готовые к применению блоки (называемые «стандартными блоками»).

Они могут поставляться на носителях данных или содержаться в библиотеках, входящих в состав пакета STEP 7 (например, IEC-функции или функции для преобразования S5/S7).

1.3 Порядок выполнения работы

1.3.1 Запустить программу SIMATIC Manager.

1.3.2 Создать новый проект, выполнив File/New… Появится окно New project. Ввести имя проекта (см. рис. 1.3).

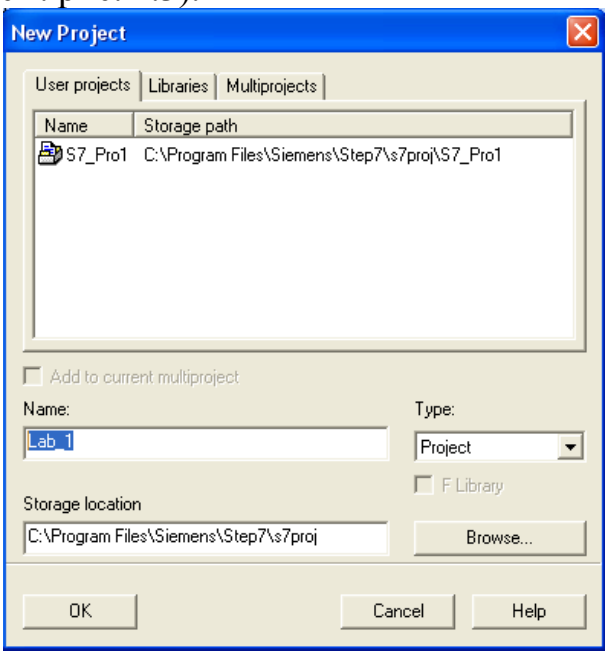


Рис. 1.3 *Окно для создания нового проекта*

1.3.3 В правой части созданного проекта кликнуть правой кнопкой мыши, в появившемся контекстном меню выбрать Insert New Object/SIMATIC 300 Station (см. рис. 1.4).

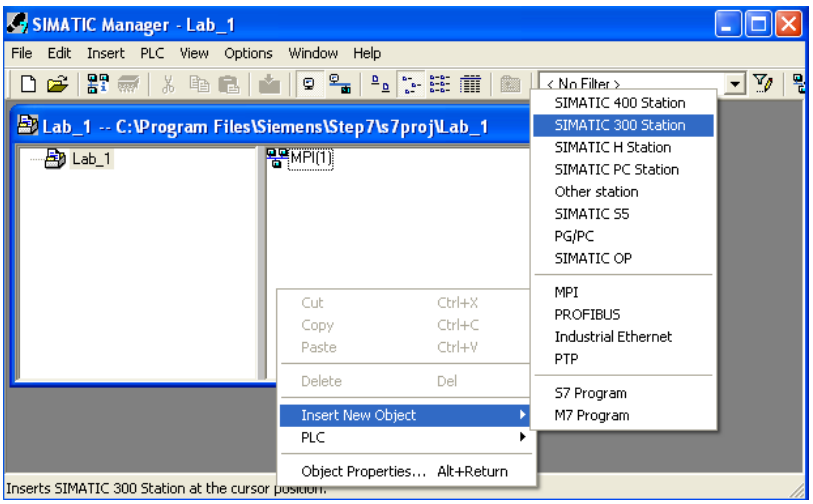


Рис. 1.4 *Выбор контроллера*

1.3.4 Открыть созданный SIMATIC 300 (1), далее открыть Hardware. В правой части появившегося окна раскрыть дерево следующим образом: SIMATIC 300/RACK-300. Перетащить элемент Rail в левую часть экрана. Аналогично перетащить в первую строку элемент PS 307 5A из PS-300; во вторую строку элемент V2.0 из CPU-300/CPU 314C-2DP; в четвёртую – V1.1 из CP-300/Industrial Ethernet/CP-343-1/6GK7 343-1EX20-0XE0 (см. рис. 1.5). Закрыть окно HW Config. Далее согласиться с сохранением, нажав Yes.

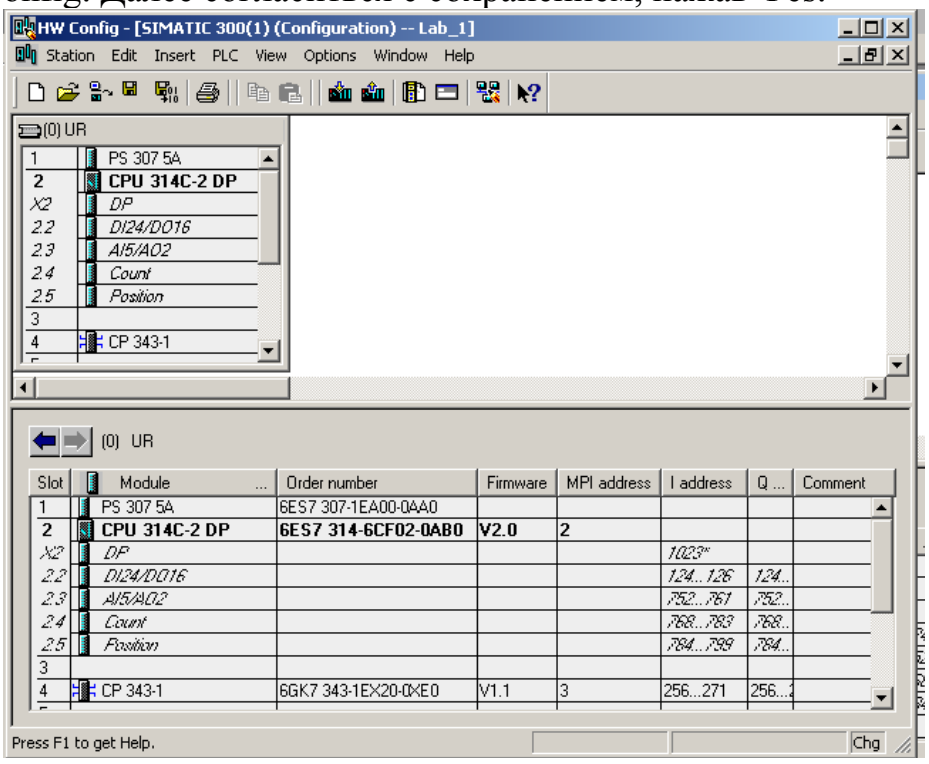


Рис. 1.5 *Конфигурация SIMATIC 300(1)*

1.3.5 В левой части окна проекта раскрыть дерево до Blocks. Открыть организационный блок OB1. В появившемся окне Properties выбрать язык, на котором будем писать программу: LAD (контактно-релейные схемы) (см. рис. 1.6).

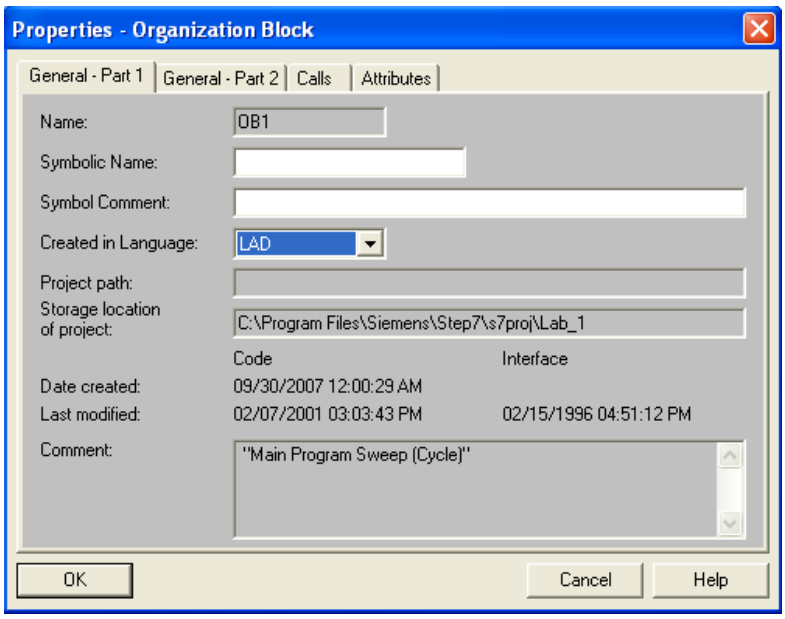


Рис. 1.6 *Окно параметров организационного блока OB1*

1.3.6 В появившемся окне написать программу выполнения поставленной задачи (снять значение температуры печи) двумя способами: 1) используя предложенный ниже алгоритм; 2) используя библиотечную функцию SCALE.

1.3.7 Алгоритм преобразования целого числа в число типа Real, для аналогового входа:

1.3.7.1 Выделив ветвь, в левой части окна из Converter, выбрать элемент I_DI (блок, преобразовывающий целое число Int в число с фиксированной точкой DInt), щёлкнув по нему два раза. Задать аналоговый вход PIW752 и выход MD1 (см. рис. 1.7).

1.3.7.2 Аналогично, выделив ветвь, добавить блок DI_R из Converter (блок, преобразовывающий число с фиксированной точкой DInt в вещественное число Real). Задать вход MD1 и выход MD1 (см. рис. 1.7).

1.3.7.3 От преобразователя приходит сигнал 0-10В, что соответствует измеряемой температуре 0-1000ºС. В АЦП контроллера аналоговый сигнал 0-10В преобразуется в цифровой, диапазон изменения которого 0-27648. Чтобы получить значение температуры, необходимо сигнал от АЦП преобразовать согласно следующей зависимости:

$$\theta = k \cdot \text{АЦП} + b,$$

где θ – температура, ºС; *АЦП* – код АЦП, в целочисленном представлении.

Для реализации данной зависимости средствами STEP 7 добавляем блок умножения MUL_R из Floating-point fct. В качестве выхода задать MD1, первого входа – MD1, а второго входа – значение коэффициента k (см. рис. 1.7).

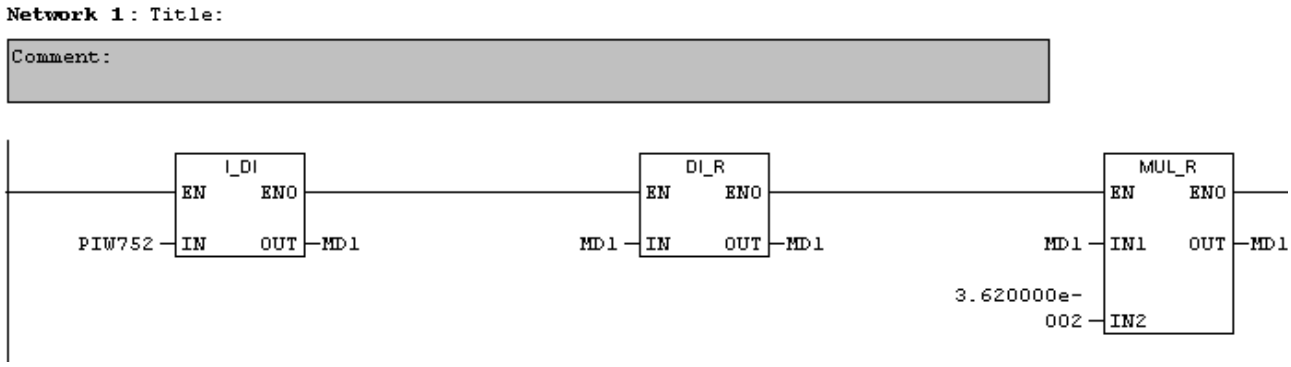


***Рис. 1.7** Первый способ выполнения программы*

Для определения коэффициентов k и b необходимо решить систему уравнений:

$$\begin{cases} \theta_{\min} = k \cdot \text{АЦП} + b \\ \theta_{\max} = k \cdot АЦП + b \end{cases}$$

1.3.8 Второй способ. Выделив ветвь, в левой части окна из Libraries/Standart Library/TI-S7 Converting Blocks выбрать FC105 SCALE Convert (см. рис. 1.8).

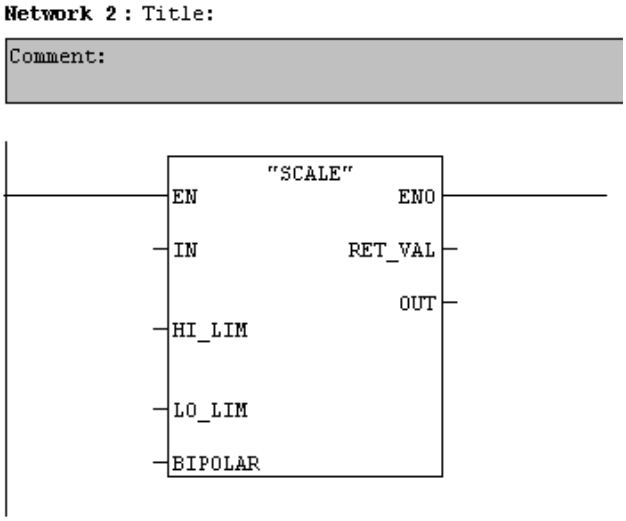


***Рис. 1.8** Второй способ выполнения программы*

Назначение входов и выходов:
IN – входной сигнал масштабируемой величины;
HI_LIM – верхний предел калибруемой величины;
L0_LIM – нижний предел калибруемой величины;
BIPOLAR – вход, задающий полярность сигнала (0 – униполярный; 1 – биполярный);
RET_VAL – возвращает 0, если при выполнении не было ошибок;
OUT – отмасштабированный выход

1.3.9 Чтобы просмотреть занятую на данный момент меркерную память, нужно выбрать в главном меню Options/Reference Data/Display и Assignment (Input, Output, Bit Memory, Timers, Counters) (см. рис. 1.9).

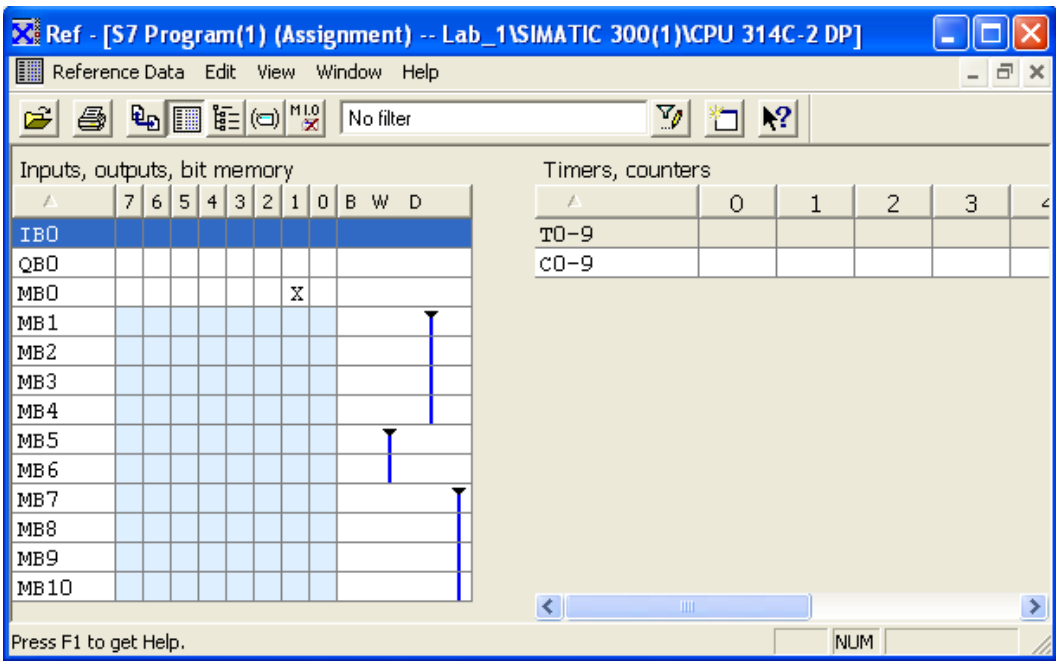


***Рис. 1.9** Окно памяти*

1.3.10 Сохранить изменения в OB1, выбрав в главном меню File/Save.
1.3.11 В левой части окна проекта сворачиваем все «-» до SIMATIC 300 (1). На панели инструментов нажать на кнопку Download  и далее согласиться со всем.
1.3.12 Запустить контроллер (установить тумблер в Run).
1.3.13 Вернуться в блок OB1 и на панели инструментов нажать на кнопку Monitor (on/off)  . Если всё выполнено правильно, то в правой части окна OB1 высвечивается результат измерения температуры печи. Для двух способов определения он должен совпадать.
1.3.14 Если необходимо отредактировать программу, надо отжать кнопку Monitor (on/off)  и выключить контроллер (тумблер в Stop). Отредактировав, повторить пункты 1.3.10-1.3.13.
1.3.15 Делать выводы по проделанной работе.

1.4 Контрольные вопросы и задания

1.4.1 Какую роль играют контроллеры в системах автоматизации?
1.4.2 Пояснить последовательность создания проекта в STEP 7.
1.4.3 На каких языках программирования возможно создавать программы в STEP 7 и в чем их особенности?
1.4.4 Какие функции выполняет блок «scale»?
1.4.5 Из каких областей состоит память контроллера SIMATIC S7-300?
1.4.6 Какие типы блоков существуют в STEP 7 и каково их функциональное назначение?

<u>Обязательные составляющие отчета</u>

1. Скриншот сконфигурированной станции S7-300 лабораторного стенда.
2. Программа на STEP 7.

Лабораторная работа №2
Визуализация и архивирование аналогового сигнала с использованием SCADA системы ProTool

2.1 Цель работы: получить навык в работе со SCADA системой ProTool.

2.2 Теоретическое введение

Что такое ProTool?

ProTool - это SCADA (Supervisory Control And Data Acquisition - диспетчерское управление и сбор данных) система. SCADA - это специализированное программное обеспечение, ориентированное на обеспечение интерфейса между диспетчером и системой управления, а также коммуникацию с внешним миром.

ProTool/Pro состоит из системы конфигурирования ProTool/Pro CS (Configuration System) и пакета для визуализации технологического процесса ProTool/Pro RT (Runtime).

ProTool/Pro предоставляет следующие ***возможности:***

- Удобную систему визуализации процесса с большим выбором стандартных элементов: полей ввода/вывода, гистограмм, графиков трендов, растровой и векторной графики и динамических атрибутов.
- Интегрированную систему сообщений
- Архивацию переменных процесса и сообщений
- Пользовательские функции на основе Visual Basic Script®
- Драйвера для связи с SIMATIC S5, SIMATIC S7, PLC других производителей.

Конфигурирование в ProTool, интегрированном в STEP 7

ProTool может быть интегрирован в конфигурационное ПО SIMATIC STEP7, позволяя таким образом использовать символы и блоки данных как теги в ProTool. Это не только экономит время, но также уменьшает возможность ошибок при повторном вводе одних и тех же данных.

Интеграция ProTool в STEP 7 дает следующие ***преимущества:***

- Можно управлять проектами ProTool, используя SIMATIC Manager (т.е. ту же программу, что и для управления проектами STEP 7).
- Можно использовать символы и блоки данных STEP 7 из таблицы символов S7 как теги. Тип данных и адрес в данном случае проставляются автоматически.
- ProTool отображает список всех PLC в проекте STEP 7 и, после выбора PLC, определяет соответствующие параметры адреса.
- В STEP 7 можно конфигурировать сообщения ALARM_S и выводить их на операторский терминал.
- Символьное имя надо назначить только один раз, и затем использовать его где угодно.

Окно проекта ProTool

При открытии нового или существующего проекта, появляется *окно проекта.*

В окне проекта *типы объектов*, которые можно конфигурировать, находятся слева, а сами объекты – справа. То, какие объекты можно конфигурировать, зависит от конкретного операторского терминала.

Различные объекты в ProTool непосредственно связаны с программами для их редактирования.

Данные проекта в ProTool хранятся в виде *объектов*. Объекты в проекте организованы в *древовидную структуру*.

Окно проекта отображает типы объектов относящихся к проекту, которые можно редактировать на данном операторском терминале. Классы объектов содержат объекты, свойства которых можно изменять.

Окно проекта организовано следующим образом (см. рис. 2.1):

- Заголовок окна содержит *имя проекта*.
- Левая половина экрана отображает *типы объектов*, которые можно конфигурировать, а содержащиеся в них объекты отображаются в правой половине.

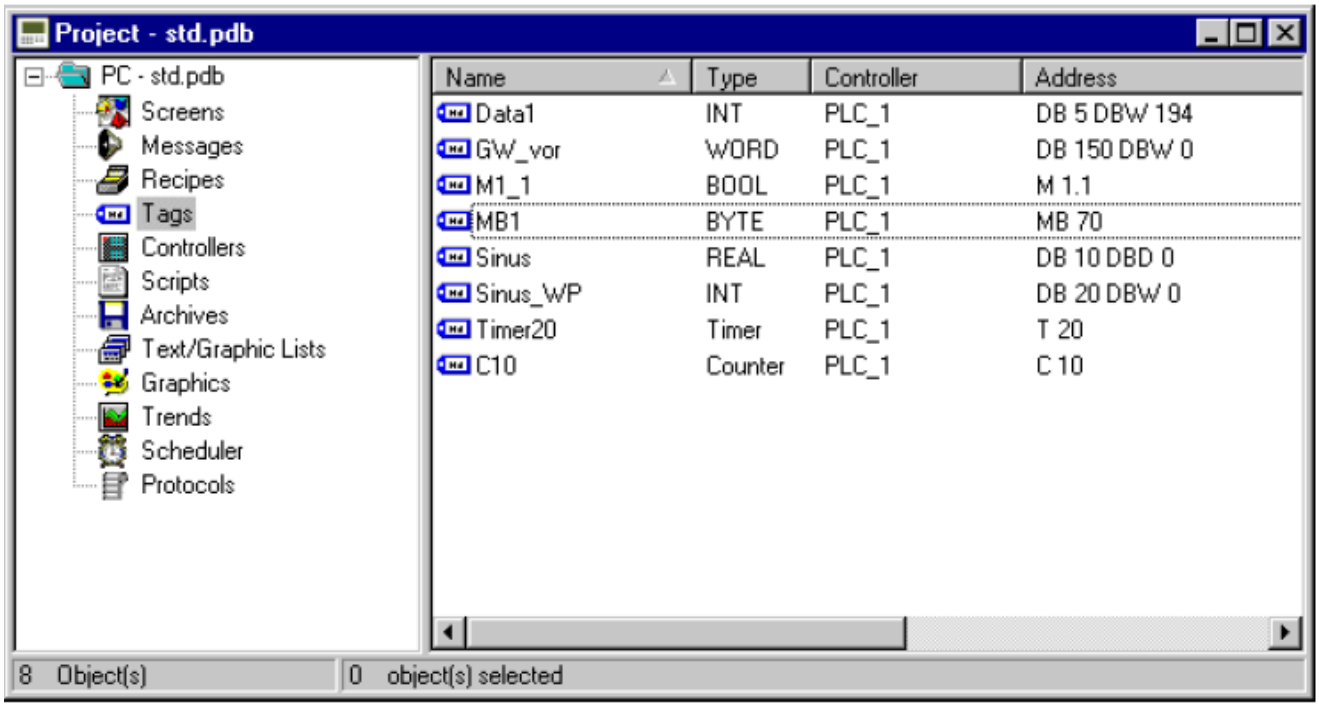


Рис. 2.1 *Пример окна проекта с тегами*

Использование тегов

Тег – это переменная SCADA системы, которая имеет символьное имя и определенный тип данных. Значение тега изменяется во время исполнения программы PLC.

Теги, связанные с PLC называются ***глобальными***. Теги, не связанные с PLC называются ***локальными***.

Глобальный тег занимает в PLC определенное адресное пространство, доступное для чтения и записи из операторского терминала и PLC.

Локальные теги доступны только в пределах операторского терминала. Локальные теги можно создавать, например, для того чтобы оператор мог вводить на операторском терминале значения уставок.

ProTool распознает следующие ***типы тегов*** (но не все из них доступны в каждом PLC):

Тип данных	Разряды	Диапазон значений
INT	16 бит	от – 32768 до 32767
UINT	16 бит	от 0 до 65535
LONG	32 бита	от – 2147483648 до 2147483647
ULONG	32 бита	от 0 до 4294967295
FLOAT	32 бита	Верхний предел: ± 3.402823 e+38 Нижний предел: ± 1.175495 e–38
BOOL	–	true (1), false (0)
STRING	–	от 1 до 255 байт
DATETIME	64 бита	Значение даты/времени
Массив тегов	Данный тип объединяет произвольное число тегов одного и того же типа в единое целое, с которым можно обращаться как с самостоятельным объектом.	

Создание архивов

Системы на базе Windows предоставляют возможность архивировать данные процесса (то есть хранить их как угодно долго и анализировать).

Можно архивировать следующие типы данных процесса:

- ***Теги.*** В диалоговом окне *Tags* (*Теги*) определяют условие активизации и диапазон значений для архивации тегов.
- ***Сообщения.*** Выбрав в меню *System→Messages→Settings* (*Система → Сообщения → Настройки*) можно определить, какие сообщения будут архивироваться.
- ***Тренды.*** Для установки архива, из которого будут читаться теги для отображения тренда, используют закладку *Data Source* (*Источники Данных*) в диалоговом окне *Trend* (*Тренд*).

Свойства архива, такие как место хранения и т.д. определяются в диалоговом окне *Archives* (*Архивы*).

Диаграмма, представленная на рис. 2.2, показывает ***модель архивирования.***

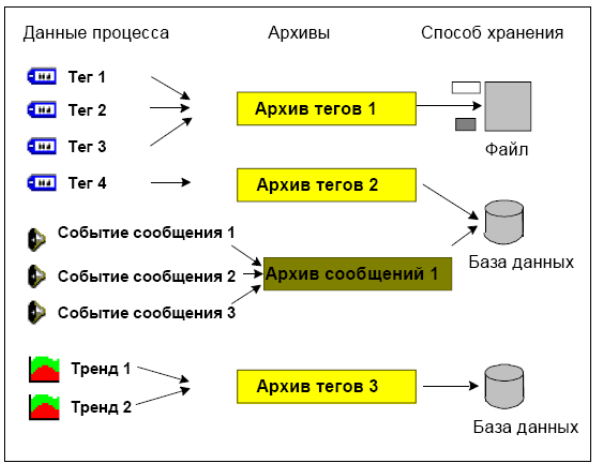


Рис. 2.2 Модель архивирования

Свойства архива. Возможны два типа архивов:

1) Short–term archive (Краткосрочный архив) – FIFO буфер, это обозначает, что если, например, буфер имеет емкость 100, то хранятся только последние 100 значений. Старые значения перезаписываются.

2) Sequence archive (Последовательный архив) – заполняется до определенного размера. Затем для продолжения работы необходимо изменить настройки архива. Для последовательных архивов можно выбрать одну из следующих опций:

- *Automatically Create Sequence Archive* (*Автоматическое Создание последовательного архива*). Архивы получают определенное имя архива добавлением номера (1 ... n). Количество архивов можно сконфигурировать. Как только последний архив оказывается заполненным, первый архив начинает заполняться вновь.
- *Output System Message When* (*Когда Выводится Системное Сообщение*). Если архив (например, дискета) заполняется, выводится системное сообщение. Можно определить уровень в процентах, при котором будет выводиться сообщение.
- *Trigger Function* (*Активизация Функции*). Когда архив полон, активизируются функции, включающие специальную обработку последовательного архива.

2.3 Порядок выполнения работы

2.3.1 Запустить программу ***SIMATIC Manager***.

2.3.2 Открыть проект, созданный в лабораторной работе №1. В левой части окна проекта раскрыть дерево до Blocks. Создать блок данных DB, щёлкнув в правой части окна правой кнопкой мыши. В появившемся контекстном меню выбрать Insert New Object/Data Block. В появившемся окне Properties (см. рис. 2.3) указать имя блока (qwer). В данном случае блок DB1 будем использовать в качестве области памяти, где будут храниться переменные.

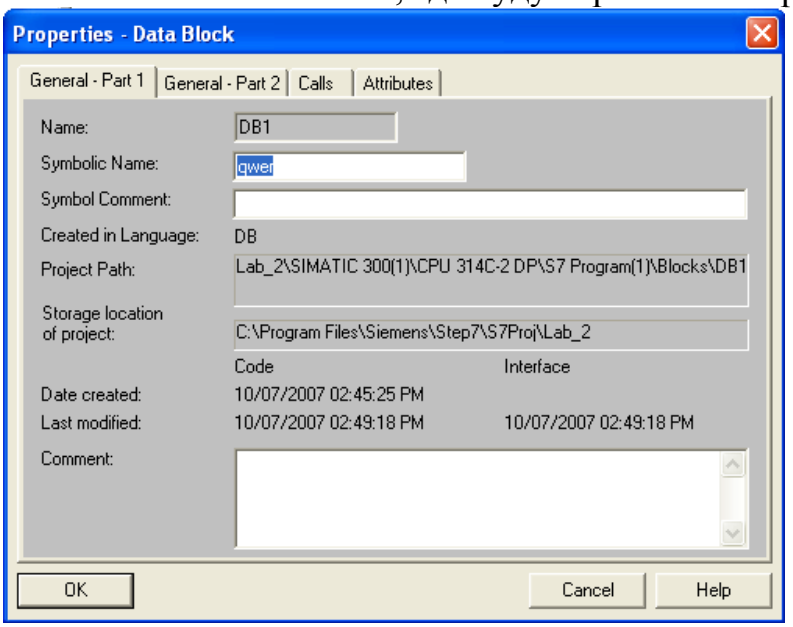


Рис. 2.3 *Окно параметров блока данных DB1*

2.3.3 Двойным щелчком открыть DB1, где создать новую переменную Temper (см. рис. 2.4). Сохранить изменения, выбрав File/Save.

DB1 -- "qwer" -- Lab_2\SIMATIC 300(1)\CPU 314C-2 DP\...\DB1

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DB_VAR	INT	0	Temporary placeholder variable
+2.0	temper	REAL	0.000000e+000	
=6.0		END_STRUCT		

Рис. 2.4 *Окно блока данных DB1*

2.3.4 Открыть OB1; в готовой программе (см. рис. 2.5) в блоке MUL_R изменить выходное значение (OUT) MD1 на *"qwer".temper* следующим образом (см. рис. 2.6): установить курсор на MD1, нажать “пробел”, раскрыть “+” для qwer и выбрать *"qwer".temper*. Сохранить изменения, выбрав File/Save.

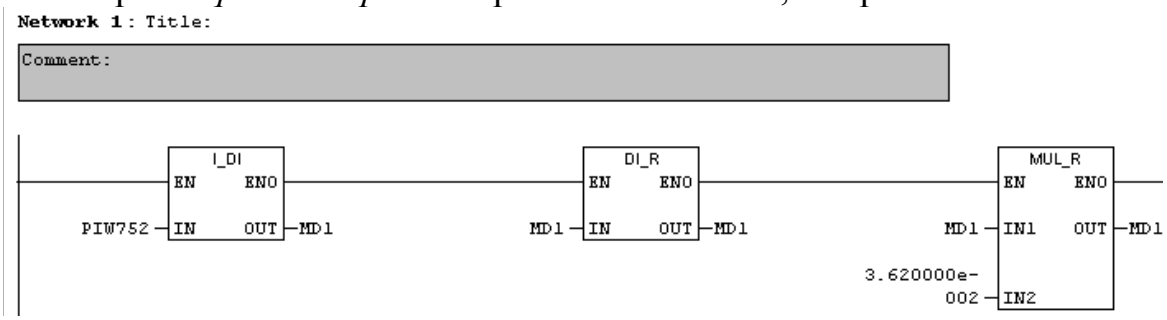


Рис. 2.5 *Исходный вид программы*

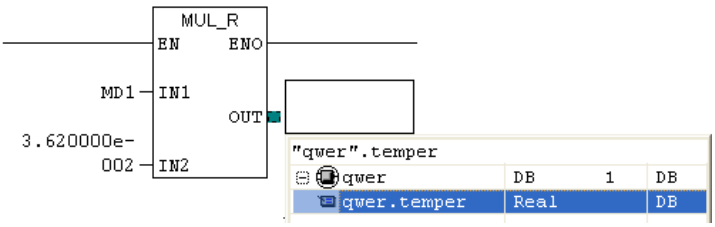


Рис. 2.6 *Назначение переменной выходу блока*

2.3.5 Запустить программу ***ProTool*** из SIMATIC/ProTool Pro CS. Создать новый проект File/New..., в появившемся окне выбрать проект, созданный в STEP7, ввести Object Name; выбрать ОК (см. рис. 2.7).

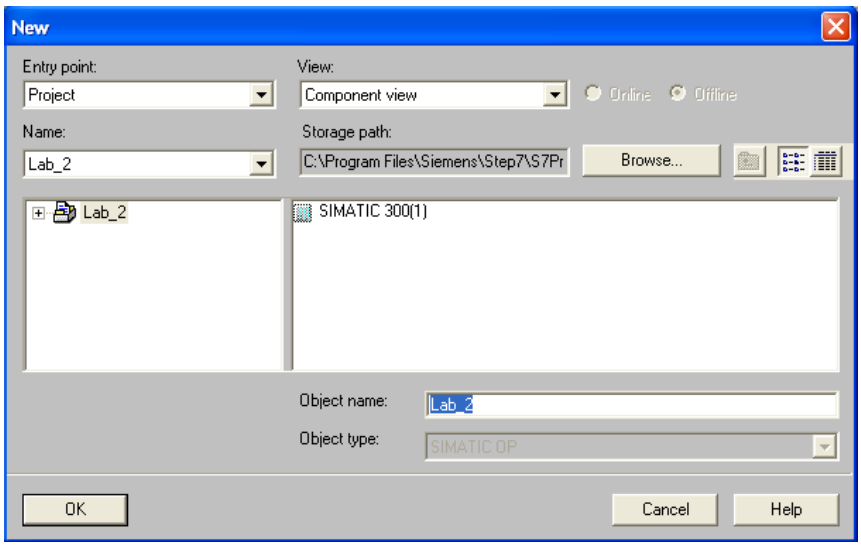


***Рис. 2.7** Окно создания нового проекта в ProTool*

2.3.6 В окне Project Wizard раскрыть "+" Windows-based systems, а затем выбрать PC и указать необходимое разрешение (см. рис. 2.8); нажать Далее >.

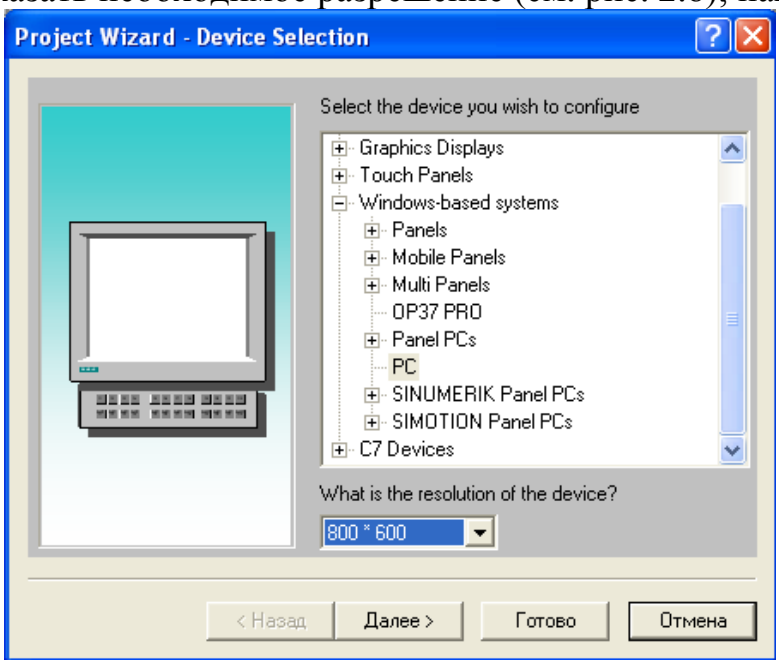


***Рис. 2.8** Окно Project Wizard - Device Selection*

2.3.7 В появившемся окне выбрать SIMATIC S7-300/400 V6.0 (см. рис. 2.9). Затем нажать Далее > и Готово.

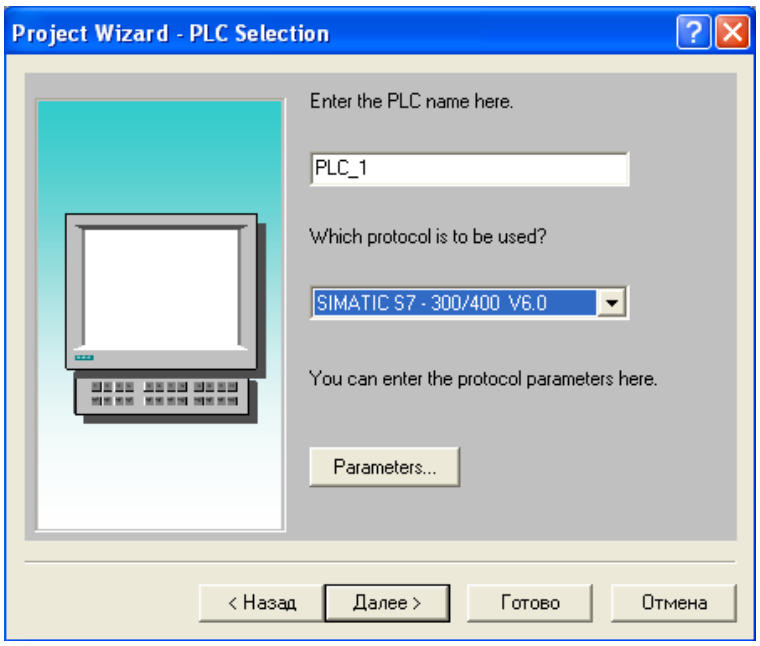


Рис. 2.9 *Окно Project Wizard – PLC Selection*

2.3.8 Создать тег, для этого в левой части окна проекта Lab_2 щёлкнуть правой кнопкой мыши по Tag, выбрать Tag insert… и в появившемся окне задать параметры тега (см. рис. 2.10). Здесь в поле DB находится имя блока данных *DB* в программе Step 7, а в поле DBD – адрес блока данных *DB* в программе Step 7. Адрес можно посмотреть в пункте 2.3.3.

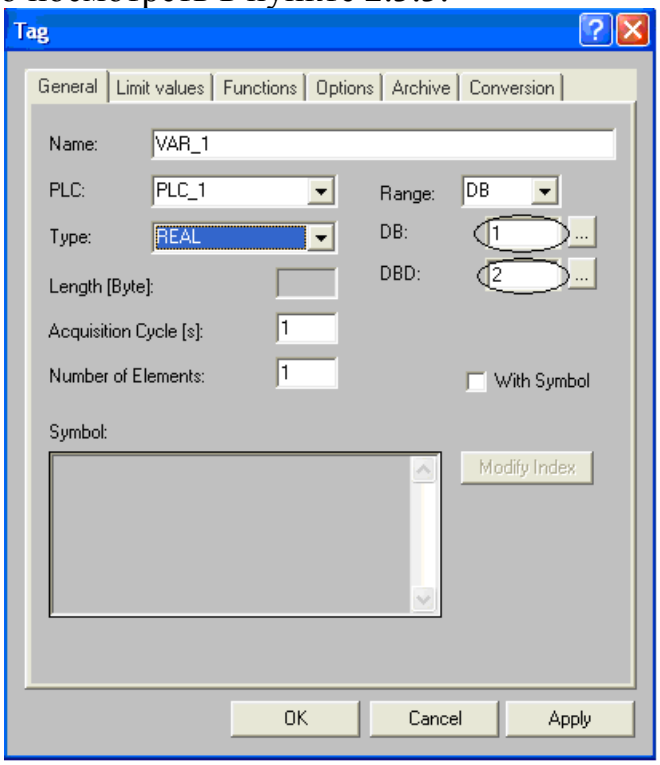


Рис. 2.10 *Окно параметров Tag*

2.3.9 В левой части окна проекта Lab_2 двойным щелчком открыть Screens. На панели инструментов выбрать Output Field  (*Поле вывода* – предназначено для вывода значений различных форматов на операторский терминал). В

появившемся окне параметров объекта Output Field задать следующее: Decimal Places: 3 (число знаков после запятой); Field Length: 9 (число знаков поля); Value: VAR_1; OK (см. рис. 2.11). В результате в рабочей области появится маска отображения результата измерения температуры печи (см. рис. 2.12). Сохранить изменения, выбрав File/Save.

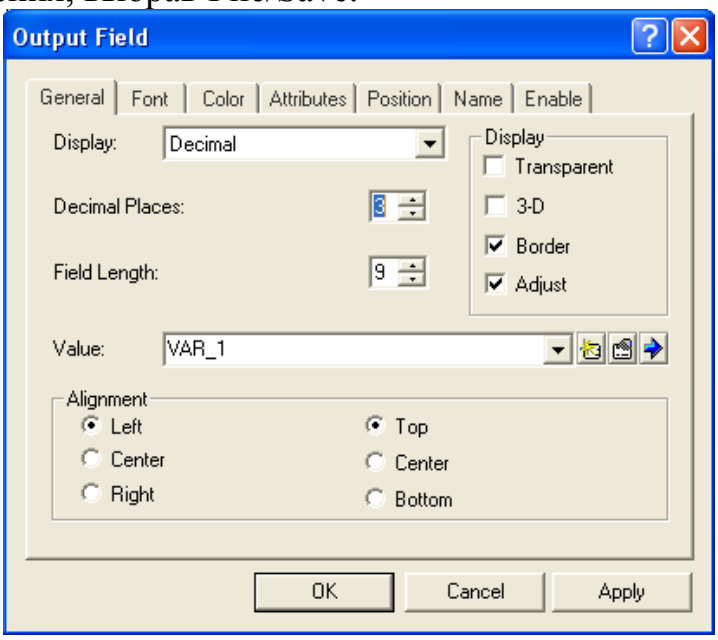


Рис. 2.11 *Окно параметров объекта Output Field*

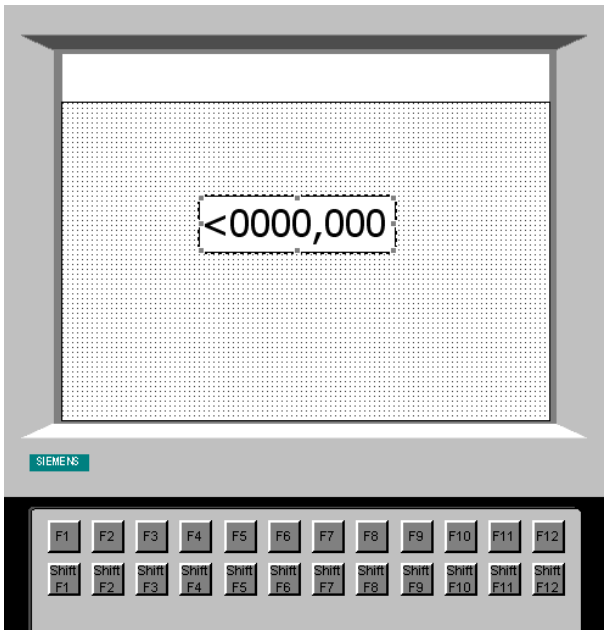


Рис. 2.12 *Рабочая область*

2.3.10 Создать архив, в котором будут сохраняться значения результата измерения температуры печи. Для этого в левой части окна проекта Lab_2 щёлкнуть правой кнопкой мыши по Archives, выбрать Archive insert… и в появившемся окне задать параметры архива (см. рис. 2.13).

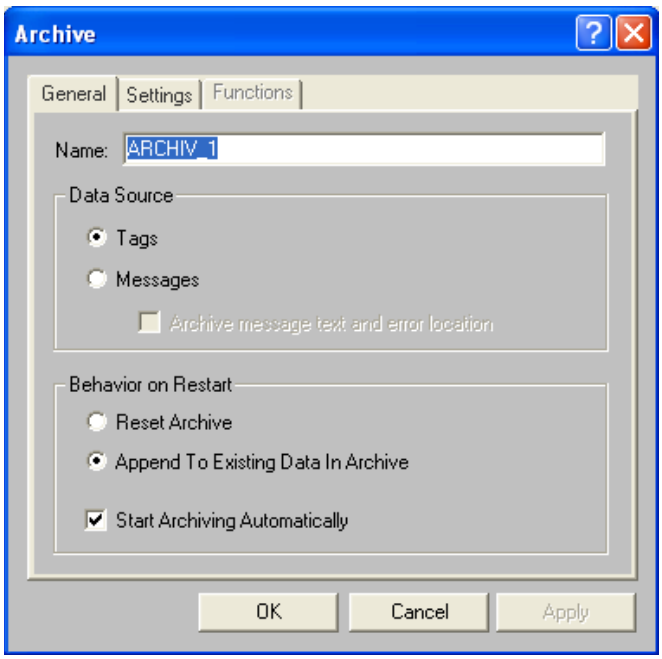


Рис. 2.13 *Окно параметров Archives*

2.3.11 Открыть окно Properties тега VAR_1, где выбрать закладку Archive и указать имя архива, в который будут записываться значения тега (в нашем случае ARCHIV_1) и установить Cyclically, равное 30s, что означает: дискретность записи данных тега в архив (см. рис. 2.14).

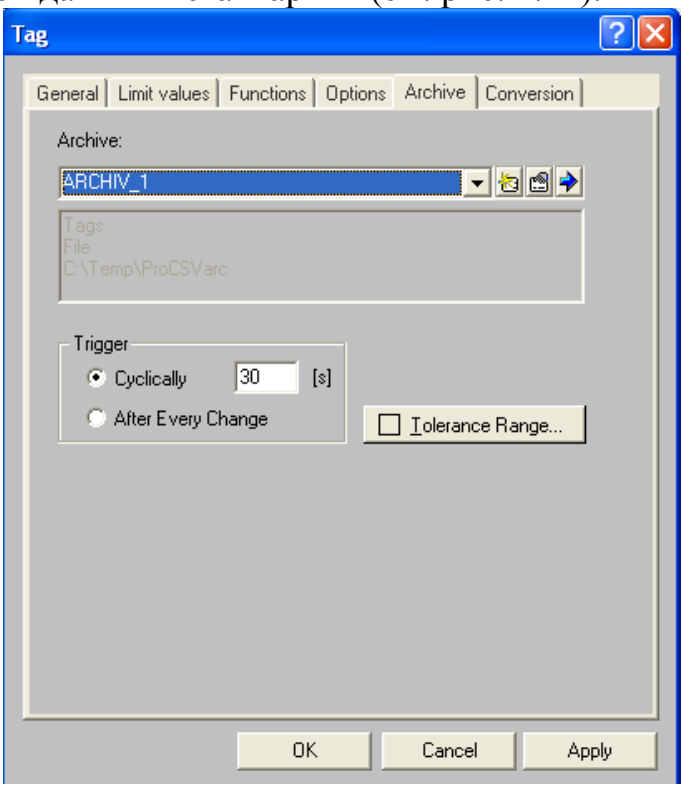


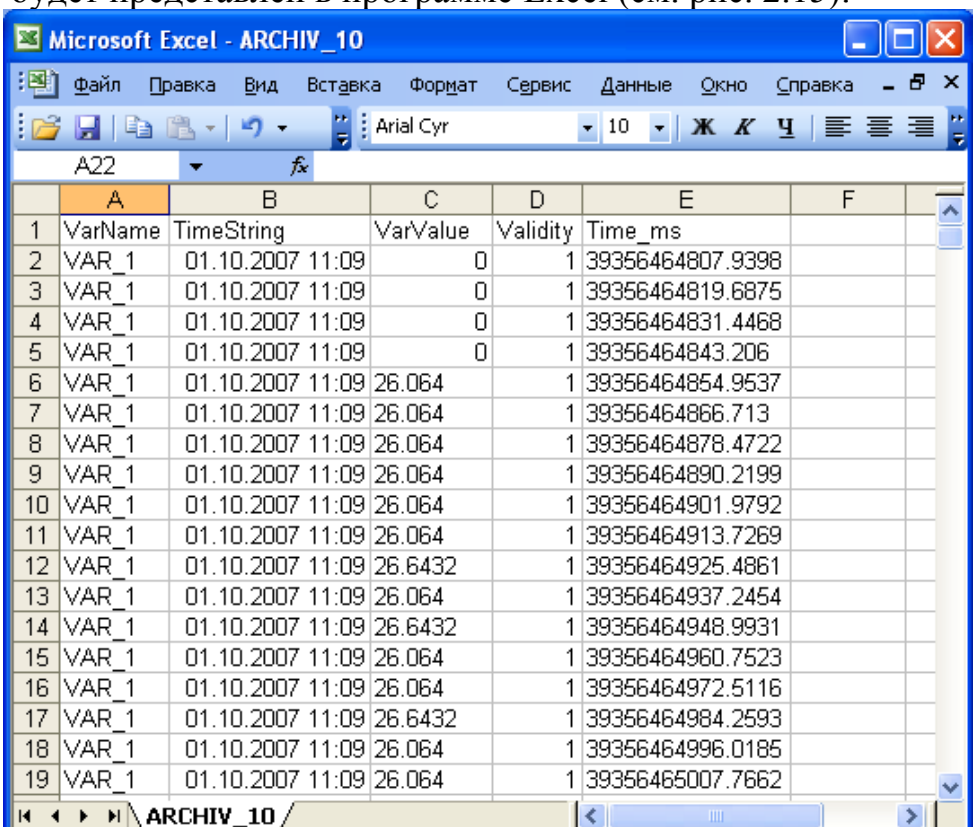
Рис. 2.14 *Окно параметров Tag*

2.3.12 Просмотреть результат выполнения проекта в режиме симуляции. Для этого на панели выбрать иконку *Start ProTool/Pro Simulator*  . В появившемся окне ввести данные в следующей последовательности:

VAR_1	REAL	0	Dec	1.0	Random		7,5	10		✓

2.3.13 Просмотреть результат выполнения проекта на реальном контроллере. В ***STEP7*** в левой части окна проекта сворачиваем все «-» до SIMATIC 300(1). На панели инструментов нажать на кнопку Download  и далее согласиться со всем. Загрузить контроллер (установить тумблер в Run). В ***ProTool*** на панели инструментов нажать на иконку *Start ProTool/Pro RT*  . Если всё выполнено правильно, то появится окно, в котором будет высвечиваться результат измерения температуры печи.

2.3.14 Просмотреть результат измерения температуры печи в Excel. Для этого в папке ProCSVate выбрать документ с именем созданного ранее архива (в данном случае *ARCHIV_10*, т.к. ProTool добавляет к имени файла символ -0-). Результат будет представлен в программе Excel (см. рис. 2.15):



	A	B	C	D	E	F
1	VarName	TimeString	VarValue	Validity	Time_ms	
2	VAR_1	01.10.2007 11:09	0	1	39356464807.9398	
3	VAR_1	01.10.2007 11:09	0	1	39356464819.6875	
4	VAR_1	01.10.2007 11:09	0	1	39356464831.4468	
5	VAR_1	01.10.2007 11:09	0	1	39356464843.206	
6	VAR_1	01.10.2007 11:09	26.064	1	39356464854.9537	
7	VAR_1	01.10.2007 11:09	26.064	1	39356464866.713	
8	VAR_1	01.10.2007 11:09	26.064	1	39356464878.4722	
9	VAR_1	01.10.2007 11:09	26.064	1	39356464890.2199	
10	VAR_1	01.10.2007 11:09	26.064	1	39356464901.9792	
11	VAR_1	01.10.2007 11:09	26.064	1	39356464913.7269	
12	VAR_1	01.10.2007 11:09	26.6432	1	39356464925.4861	
13	VAR_1	01.10.2007 11:09	26.064	1	39356464937.2454	
14	VAR_1	01.10.2007 11:09	26.6432	1	39356464948.9931	
15	VAR_1	01.10.2007 11:09	26.064	1	39356464960.7523	
16	VAR_1	01.10.2007 11:09	26.064	1	39356464972.5116	
17	VAR_1	01.10.2007 11:09	26.6432	1	39356464984.2593	
18	VAR_1	01.10.2007 11:09	26.064	1	39356464996.0185	
19	VAR_1	01.10.2007 11:09	26.064	1	39356465007.7662	

Рис. 2.15 *Отображение результатов работы печи в программе Excel*

2.3.15 Просмотреть результат измерения температуры печи в MatLab. Для этого надо запустить MatLab и открыть документ ARCHIV_10, выбрав File/Open… Результат будет представлен в программе MatLab (см. рис. 2.16):

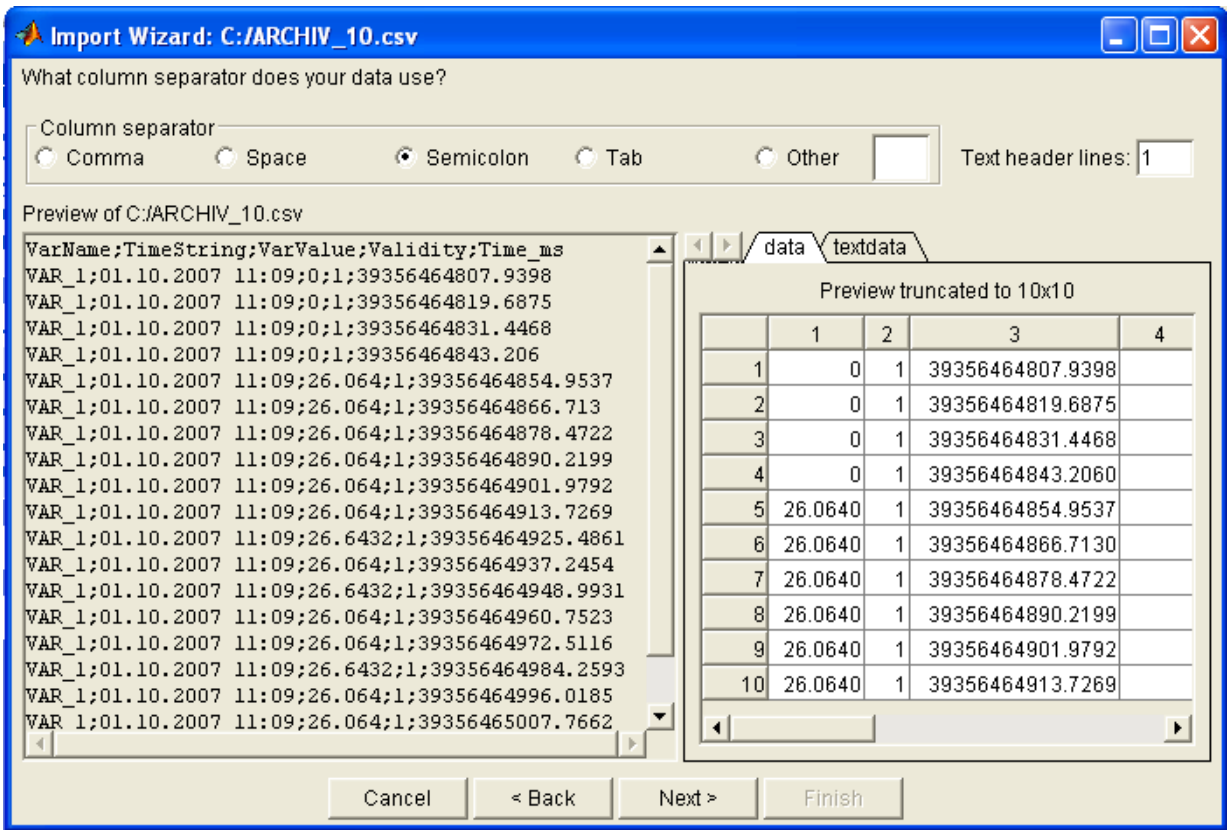


Рис. 2.16 *Отображение результатов работы печи в программе MatLab*

2.3.16 Сделать выводы по проделанной работе.

2.4 Контрольные вопросы и задания

2.4.1 Что такое SCADA система?
2.4.2 Поясните принцип конфигурирования адреса тега.
2.4.3 Какие типы архивов поддерживаются SCADA системой Protool?
2.4.4 Что такое тег, какие типы тегов вы знаете?
2.4.5 Каким образом конфигурируется архив в SCADA системе Protool?

Обязательные составляющие отчета

1. Программа на STEP 7.
2. Скриншоты разработки проекта в SCADA системе Protool.

Лабораторная работа №3
Реализация ШИМ в STEP7 и организация съема переходной характеристики лабораторной печи

3.1 Цель работы: ознакомиться с принципом реализации ШИМ в STEP7.

3.2 Теоретическое введение

Широтно-импульсная модуляция (ШИМ)

Используя широтно-импульсную модуляцию, осуществляется преобразование аналогового значения управляющей переменной *u(t)* в последовательность импульсов «имп» с периодом $T_{период}$ (см. рис. 3.1).

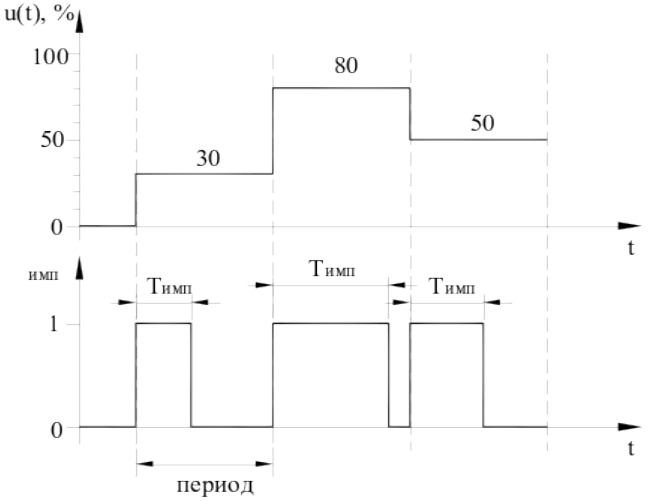


Рис. 3.1 *Временная диаграмма ШИМ*

Обзор инструкций

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).

Нормально открытый контакт (Адрес)

Обозначение:

```
<адрес>
---| |---
```

Параметр	*Тип данных*	*Область памяти*	*Описание*
<адрес>	BOOL	I,Q,M,L,D,T,C	Адрес опрашиваемого бита

Описание:

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 равен 1. Если состояние сигнала по указанному адресу равно 0, то контакт разомкнут, и команда дает результат логической операции равный 0.

Нормально замкнутый контакт (Адрес)

Обозначение:

<адрес>

---| / |---

Параметр	*Тип данных*	*Область памяти*	*Описание*
<адрес>	BOOL	I,Q,M,L,D,T,C	Адрес опрашиваемого бита

Описание:

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 равен 1. Если состояние сигнала по указанному адресу равно 1, то контакт разомкнут, и команда дает результат логической операции равный 0.

Инверсия результата логической операции

Обозначение:

---|NOT|---

Описание:

Инструкция *инверсия результата логической операции* выполняет изменение на противоположное значение результата логической операции.

Выходная катушка

Обозначение:

<адрес>

---( )

Параметр	*Тип данных*	*Область памяти*	*Описание*
<адрес>	BOOL	I,Q,M,L,D	Управляемый бит

Описание:

Выходная катушка работает как катушка в цепи управления релейно-контактной схемы. Если к катушке подводится ток, бит <адрес> устанавливается в «1». Если к катушке не подводится ток, бит <адрес> устанавливается в «0». Выходную катушку можно установить только на правом конце логической цепи. Возможно использование нескольких выходных катушек (максимум 16).

Выделение отрицательного фронта логической операции

Обозначение:

<адрес>
---(N)---

Параметр	*Тип данных*	*Область памяти*	*Описание*
<адрес>	BOOL	I,Q,M,L,D	Адрес указывает, какой бит памяти будет хранить RLO предыдущего цикла

Описание:

Инструкция *Выделение отрицательного фронта RLO* обнаруживает изменение с 1 на 0 (падающий фронт) по указанному адресу и отображает это установкой RLO в 1 после выполнения инструкции. Текущее состояние RLO сравнивается с состоянием сигнала операнда (бит памяти фронта). Если состояние сигнала операнда равно 1, а RLO перед выполнением инструкции равен 0, то RLO после выполнения инструкции будет равен 1 (импульс). Во всех остальных случаях RLO равен 0. Входной RLO затем сохраняется в указанном бите памяти.

Выделение положительного фронта логической операции

Обозначение:

<адрес>
---(P)---

Параметр	*Тип данных*	*Область памяти*	*Описание*
<адрес>	BOOL	I,Q,M,L,D	Адрес указывает, какой бит памяти будет хранить RLO предыдущего цикла

Описание:

Инструкция *Выделение положительного фронта RLO* обнаруживает изменение с 0 на 1 (нарастающий фронт) по указанному адресу и отображает это с помощью значения RLO, равного 1, после выполнения инструкции. Текущее состояние RLO сравнивается с состоянием сигнала операнда (бит памяти фронта). Если состояние сигнала операнда равно 0, а RLO перед выполнением инструкции равен 1, то RLO будет равен 1 (импульс) после выполнения инструкции. Во всех остальных случаях RLO равен 0. Входной RLO затем сохраняется в указанном бите памяти.

Таймеры

Основные понятия

Таймеры (Timers) – это ячейки памяти, используемые для реализации интервалов ожидания и мониторинга.

Таймеры позволяют программно реализовать последовательности синхронизации, такие как интервалы ожидания и наблюдения, измерение интервалов или генерирование импульсов.

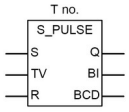
Существуют следующие ***типы таймеров***:

1) Импульсные таймеры (Pulse timers);
2) Импульсные таймеры с памятью (Extended pulse timers);
3) Таймеры задержки включения (On-delay timers);
4) Таймеры задержки включения с запоминанием (Retentive on-delay timers);
5) Таймеры задержки выключения (Off-delay timers).

Таймер	***Описание***
S_PULSE Таймер «Импульс»	Максимальное время в течение которого выходной сигнал остается равным 1, совпадает с запрограммированным временем T. Выход сбрасывается раньше, если входной сигнал меняет состояние на 0.
S_PEXT Таймер «Импульс с памятью»	Выходной сигнал остается равным 1 в течение запрограммированного времени независимо от того, как долго остается равным 1 входной сигнал.
S_ODT Таймер «Задержка включения»	Выходной сигнал устанавливается в 1 только по истечении запрограммированного времени, при этом входной сигнал все еще должен быть равен 1.
S_ODTS Таймер «Задержка включения с памятью»	Выходной сигнал устанавливается в 1 только по истечении запрограммированного времени независимо от того, как долго остается равным 1 входной сигнал.
S_OFFDT Таймер «Задержка выключения»	Выходной сигнал устанавливается в 1, когда устанавливается в 1 входной сигнал, и остается равным 1, пока таймер работает. Отсчет времени начинается, когда входной сигнал меняется с 1 на 0.

Далее рассмотрим только ***S_PULSE*** и ***S_PEXT***, которые используются при выполнении лабораторной работы.

S_PULSE: Задание параметров и запуск таймера «Импульс»
Обозначение:



Параметр	*Тип данных*	*Область памяти*	*Описание*
T no.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU
S	BOOL	I,Q,M,D,L	Вход запуска
TV	S5TIME	I,Q,M,D,L	Установка времени (от 0-9990)
R	BOOL	I,Q,M,D,L	Вход сброса

BI	WORD	I,Q,M,D,L	Остаток времени (значение в двоичном коде)
BCD	WORD	I,Q,M,D,L	Остаток времени (значение в формате BCD)
Q	BOOL	I,Q,M,D,L	Состояние таймера

Описание:

S_PULSE: (S5 таймер «Импульс») запускает заданный таймер по нарастающему фронту (изменение состояния сигнала с 0 на 1) на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, пока состояние сигнала на входе S остается равным 1. Пока таймер работает, опрос выхода Q на высокий уровень дает результат логической операции 1. Если на входе S сигнал меняется с 1 на 0 до истечения заданного времени, таймер останавливается. Тогда опрос состояния сигнала на 1 на выходе Q дает 0. Если во время работы таймера происходит изменение с 0 на 1 сигнала на входе сброса (R), то таймер сбрасывается. Это изменение сбрасывает в ноль время и базу времени. Единица на входе R таймера не оказывает никакого влияния если таймер не работает.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на выходе BI представлено в двоичном формате, а на BCD – в двоично-десятичном формате. Текущее время равно разнице между начальным значением, заданным на входе TV и временем, прошедшим с момента запуска таймера.

Временные диаграммы (см. рис. 3.2):

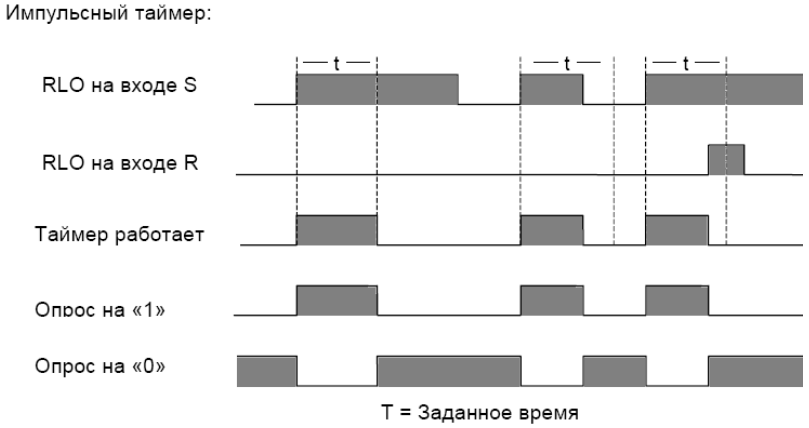
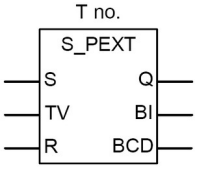


Рис. 3.2 Временные диаграммы таймера S_PULSE

S_PEXT: Задание параметров и запуск таймера «Импульс с памятью»

Обозначение:



Параметр	*Тип данных*	*Область памяти*	*Описание*
T no.	TIMER	T	Номер таймера. Диапазон номеров зависит от CPU
S	BOOL	I,Q,M,D,L	Вход запуска
TV	S5TIME	I,Q,M,D,L	Установка времени
R	BOOL	I,Q,M,D,L	Вход сброса
BI	WORD	I,Q,M,D,L	Остаток времени (значение в двоичном коде)
BCD	WORD	I,Q,M,D,L	Остаток времени (значение в формате BCD)
Q	BOOL	I,Q,M,D,L	Состояние таймера

Описание:

S_PEXT:(S5 таймер «Удлиненный Импульс») запускает заданный таймер, по нарастающему фронту на входе запуска (S). Для запуска таймера всегда необходимо изменение сигнала. Таймер продолжает работать в течение времени, заданного на входе TV, даже если состояние сигнала на входе S меняется на 0 до истечения заданного времени. Пока таймер работает выход Q выдает сигнал 1. Таймер перезапускается с заданным временем, если состояние сигнала на входе S меняется с 0 на 1 во время работы таймера.

Если во время работы таймера происходит изменение с 0 на 1 сигнала на входе сброса (R), то таймер сбрасывается. Это изменение сбрасывает в ноль время и базу времени.

Текущее значение времени может быть считано на выходах BI и BCD. Значение времени на BI представлено в двоичном формате, а на BCD – в двоично-десятичном формате. Текущее время равно разнице между начальным значением, заданным на входе TV и временем, прошедшим с момента запуска таймера.

Временные диаграммы (см. рис. 3.3):

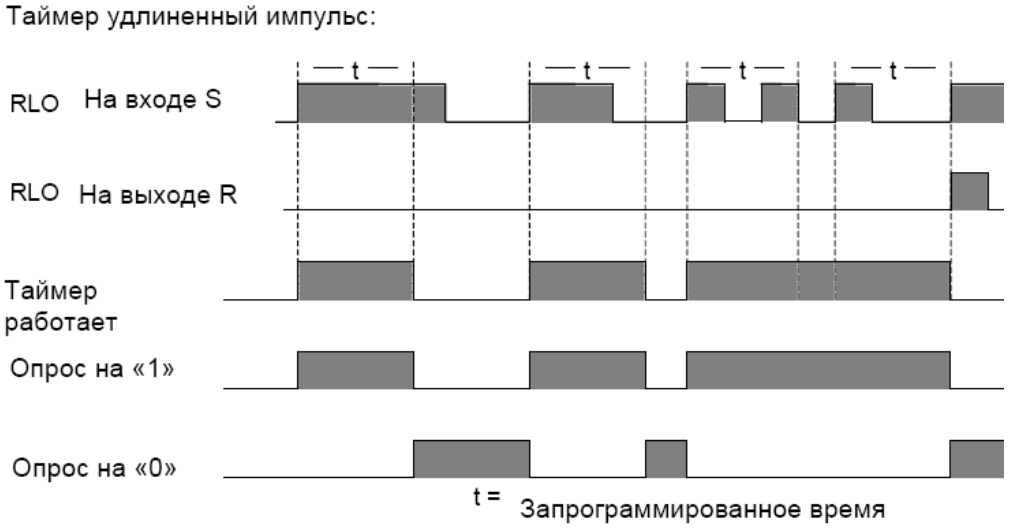


Рис. 3.3 Временные диаграммы таймера S_PEXT

Компоненты таймера

Значение времени:

Биты с 0 по 9 в таймерном слове содержат значение времени в двоичном коде. Значение времени задает количество временных отрезков. Когда таймер актуализируется, значение времени уменьшается на одну единицу через интервалы, установленные базой времени. Значение времени уменьшается до тех пор, пока оно не станет равным нулю. Задавать значение времени можно в двоичном, шестнадцатиричном или двоично-десятичном коде (BCD).

Загрузить значение времени можно с использованием следующего синтаксиса:

- S5T#aH_bM_cS_dMS,

где: a = часы,
b = минуты,
c = секунды,
d = миллисекунды.

Максимальное время, которое можно ввести, составляет 9 990 секунд или 2H_46M_30S.

S5TIME#4S = 4 секунды
s5t#2h_15m = 2 часа и 15 минут
S5T#1H_12M_18S = 1 час, 12 минут и 18 секунд

3.3 Порядок выполнения работы

3.3.1 Запустить программу ***SIMATIC Manager***.

3.3.2 Создать новый проект и сконфигурировать контроллер в STEP7.

3.3.3 В левой части окна проекта раскрыть дерево до Blocks. Открыть организационный блок OB1. В появившемся окне Properties выбрать язык, на котором будем писать программу: LAD (контактно-релейные схемы) (см. рис. 3.4).

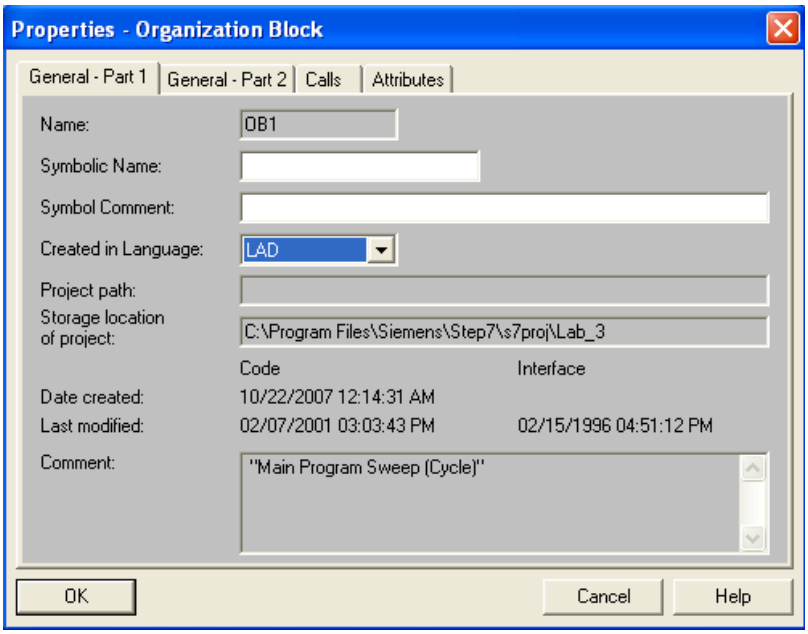


Рис. 3.4 *Окно параметров организационного блока OB1*

3.3.4 В появившемся окне написать программу выполнения поставленной задачи (реализовать съём переходной характеристики) (см. рис. 3.5):

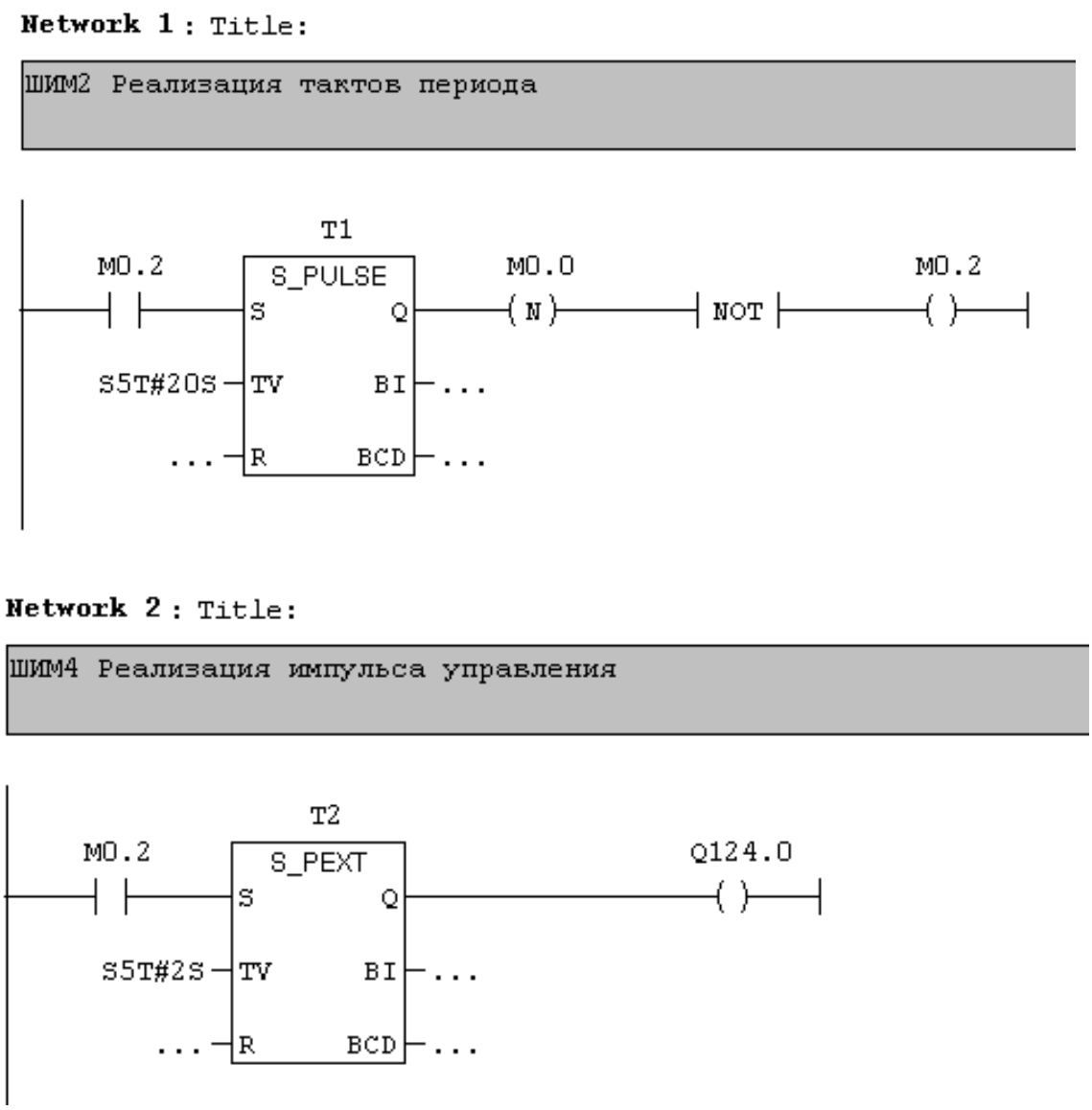


Рис. 3.5 *Программа выполнения поставленной задачи*

3.3.5 Сохранить изменения в OB1, выбрав в главном меню File/Save.
3.3.6 В левой части окна проекта сворачиваем все «-» до SIMATIC 300(1). На панели инструментов нажать на кнопку Download  и далее согласиться со всем.
3.3.7 Загрузить контроллер или Simulating (установить тумблер в Run).
3.3.8 Вернуться в блок OB1 и на панели инструментов нажать на кнопку Monitor (on/off)  . Если всё выполнено правильно, то программа будет работать как показано на рис. 3.6, 3.7.

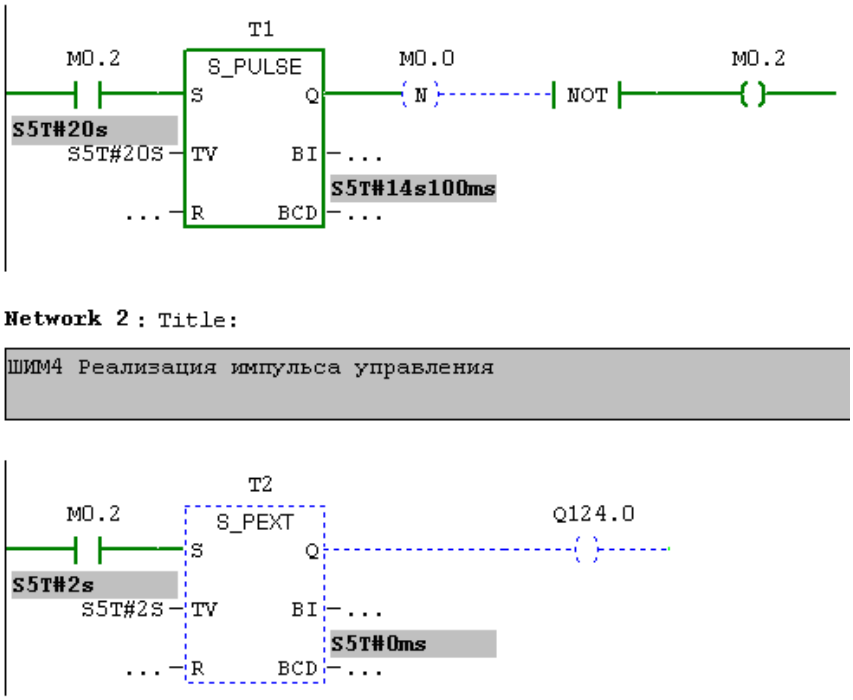


Рис. 3.6 *Считает таймер T1*

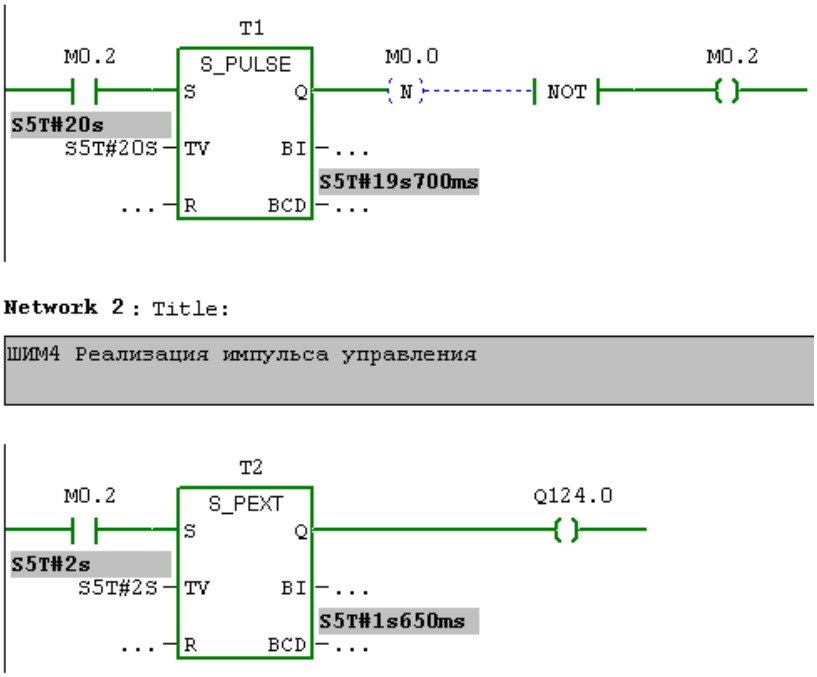
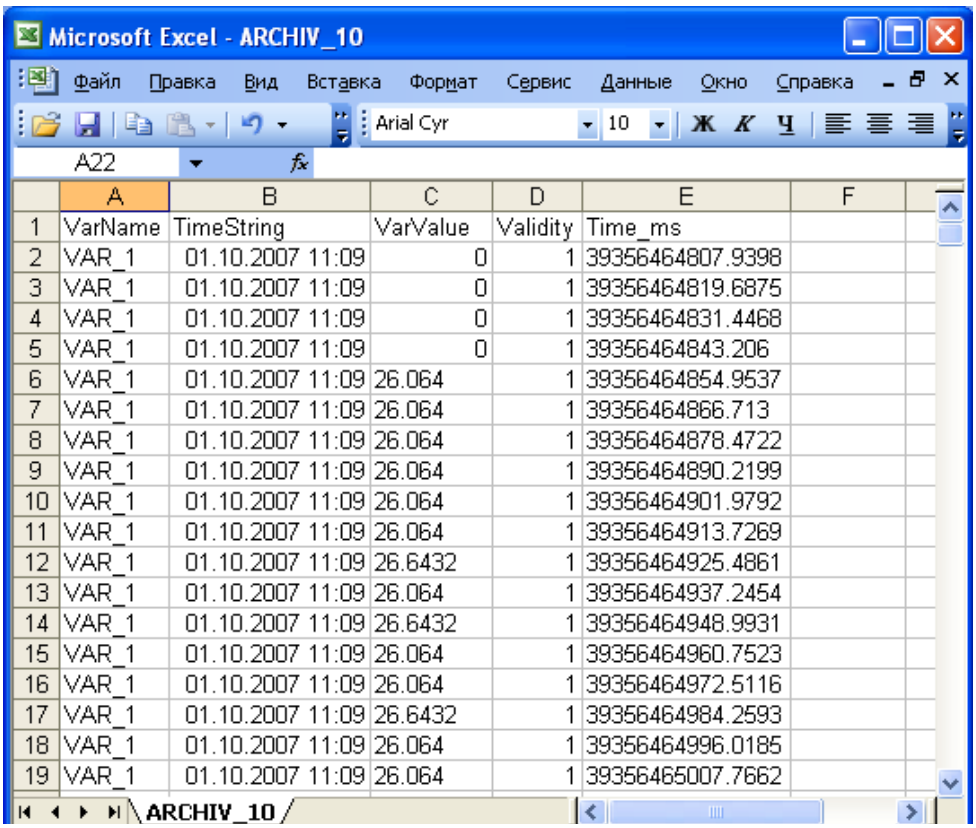


Рис. 3.7 *Считают таймеры T1 и T2*

3.3.9 Если необходимо отредактировать программу, надо отжать кнопку Monitor (on/off)  и выключить контроллер или Simulating (установить тумблер на STOP). Отредактировав, повторить пункты 3.3.8-3.3.10.

3.3.10 Создать архив, используя ProTool (см. рис. 3.8). Принцип формирования архива описан в лабораторной работе №2.



Microsoft Excel - ARCHIV_10

Файл Правка Вид Вставка Формат Сервис Данные Окно Справка

Arial Cyr 10

A22

	A	B	C	D	E	F
1	VarName	TimeString	VarValue	Validity	Time_ms	
2	VAR_1	01.10.2007 11:09	0	1	39356464807.9398	
3	VAR_1	01.10.2007 11:09	0	1	39356464819.6875	
4	VAR_1	01.10.2007 11:09	0	1	39356464831.4468	
5	VAR_1	01.10.2007 11:09	0	1	39356464843.206	
6	VAR_1	01.10.2007 11:09	26.064	1	39356464854.9537	
7	VAR_1	01.10.2007 11:09	26.064	1	39356464866.713	
8	VAR_1	01.10.2007 11:09	26.064	1	39356464878.4722	
9	VAR_1	01.10.2007 11:09	26.064	1	39356464890.2199	
10	VAR_1	01.10.2007 11:09	26.064	1	39356464901.9792	
11	VAR_1	01.10.2007 11:09	26.064	1	39356464913.7269	
12	VAR_1	01.10.2007 11:09	26.6432	1	39356464925.4861	
13	VAR_1	01.10.2007 11:09	26.064	1	39356464937.2454	
14	VAR_1	01.10.2007 11:09	26.6432	1	39356464948.9931	
15	VAR_1	01.10.2007 11:09	26.064	1	39356464960.7523	
16	VAR_1	01.10.2007 11:09	26.064	1	39356464972.5116	
17	VAR_1	01.10.2007 11:09	26.6432	1	39356464984.2593	
18	VAR_1	01.10.2007 11:09	26.064	1	39356464996.0185	
19	VAR_1	01.10.2007 11:09	26.064	1	39356465007.7662	

ARCHIV_10

Рис. 3.8 *Отображение результатов в программе Excel*

3.3.11 Построить график кривой переходного процесса (см. рис. 3.9). Это можно реализовать с помощью средств пакетов прикладных программ Microsoft Excel или Matlab.

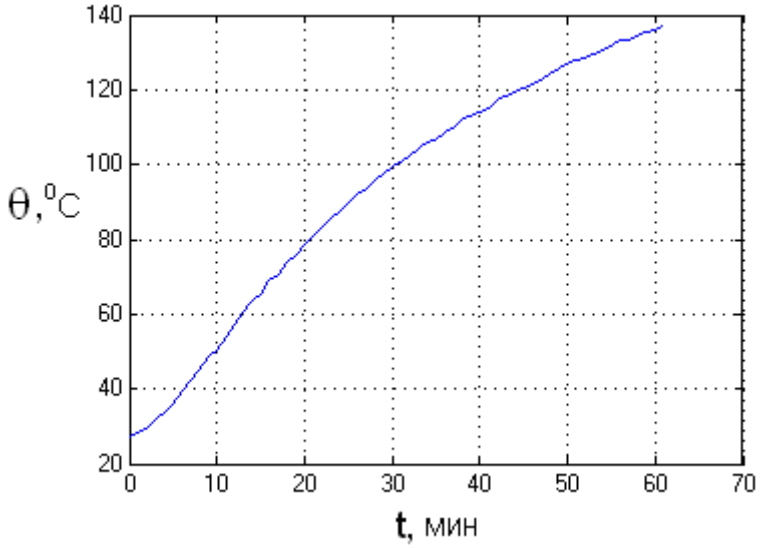


Рис. 3.9 *График кривой переходного процесса*

3.3.12 Сделать выводы по проделанной работе.

3.4 Контрольные вопросы и задания

3.4.1 Что такое ШИМ и каков принцип данного вида модуляции непрерывного сигнала?

3.4.2 Для чего используется инструкция «выходная катушка» в STEP 7?

3.4.3 Поясните принцип работы инструкции «выделение положительного фронта логической операции».
3.4.4 Поясните принцип работы таймеров «S_PULSE» и «S_PEXT».

Обязательные составляющие отчета

1. Программа ШИМ на STEP 7.
2. Архив значений переходного процесса.
3. График переходного процесса.

Лабораторная работа №4
Идентификация объекта управления

4.1 Цель работы: освоение методики идентификации объекта управления.

4.2 Теоретическое введение

Задача идентификации объекта управления заключается в нахождении его математической модели, которая в некотором смысле наилучшим способом описывает его динамические свойства. Если конечной целью является нахождение оптимального управления, то достаточно, чтобы при заданном входе выход модели был эквивалентен выходу системы. В этом случае нет необходимости в том, чтобы структура и параметры модели совпадали со структурой и параметрами физической системы.

Вообще же задача построения математической модели заключается в определении структуры объекта, нахождении значений параметров и, если необходимо, значений зависимых переменных, например, переменных состояния.

Методы идентификации в большинстве случаев при заданной ограниченной точности измерений не позволяют построить сложную модель, эквивалентную по структуре и параметрам реальному объекту. Этот факт, однако, не мешает последующему использованию такой модели, если, конечно, она отражает существенные стороны объекта. Более того, именно в силу своей простоты такая модель наиболее пригодна для последующего использования.

Идентификацию можно провести либо методами физико-математического анализа, либо методами экспериментального анализа.

При идентификации методами физико-математического анализа исходят из конструктивных данных и математического описания основных процессов, которые имеют место в изучаемом объекте. Получают систему алгебраических и дифференциальных уравнений, содержащих как входные и выходные переменные, так и переменные состояния. В эти уравнения иногда включаются избыточные внутренние переменные объекта, которые можно не учитывать.

При идентификации методами экспериментального анализа обычно находят математическую модель устойчивого объекта по измерениям его входных и выходных величин.

В работе исследуется система, состоящая из двух тепловых объектов – это печь и термопара. Каждый из этих объектов можно описать математически апериодическим звеном первого порядка и звеном запаздывания, следовательно, общую модель системы следует искать в виде:

$$W_{об}(s) = \frac{1}{T_1 \cdot s + 1} \cdot \frac{K_{об}}{T_2 \cdot s + 1} \cdot e^{(-\tau_{об} \cdot s)}$$

Для таких объектов при внесении возмущения в виде единичного скачка может получиться переходная функция, показанная на рис. 4.1. Она представляет собой монотонную кривую, характерной точкой которой является точка перегиба, соответствующая моменту изменения знака второй производной.

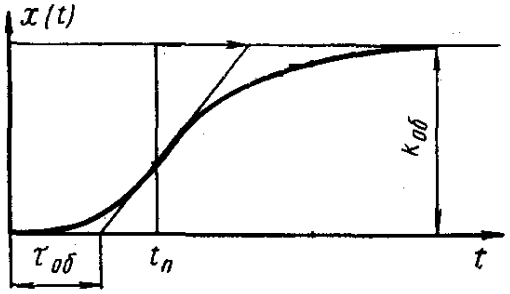


Рис. 4.1 *Переходная характеристика статического объекта второго порядка*

Передаточная функция такого объекта может рассматриваться как произведение передаточных функций двух апериодических объектов с постоянными времени T_1, и T_2. Переходная функция определяется выражением:

$$Y_{мод}(t_i) = y(0) + K_{об}\left(1 - \frac{T_1}{T_1 - T_2} \cdot e^{\frac{-(t_i - \tau_{об})}{T_1}} + \frac{T_2}{T_1 - T_2} \cdot e^{\frac{-(t_i - \tau_{об})}{T_2}}\right)$$

Для приближенного определения динамических параметров статического объекта (запаздывания $\tau_{об}$, коэффициента передачи $K_{об}$) в точке перегиба изменения выходной величины проводят касательную к переходной характеристике и продолжают ее до пересечения с линией начального значения выходной величины (осью абсцисс). Отрезок времени от момента внесения возмущения до точки пересечения касательной с осью определит запаздывание объекта $\tau_{об}$.

Коэффициент передачи статического объекта представляет собой изменение выходной величины объекта при переходе из начального в новое установившееся состояние, отнесенное к изменению возмущения на входе:

$$K_{об} = \frac{x_\infty - x_0}{\Delta x_{вх}},$$

где x_0 – значение выходной величины в начальном установившемся состоянии;

x_∞ – то же, для нового установившегося состояния;

$\Delta x_{вх}$ – величина вносимого возмущения.

Для определения значений постоянных времени можно воспользоваться различными методами оптимизации, такими как градиентный метод, покоординатного спуска, простого перебора значений с достижением минимума функционала, представляющего собой сумму квадратов невязок:

$$F = \sum_{i=1}^{N} (Y_{мод}(t_i) - Y_{эксп}(t_i))^2$$

4.3 Порядок выполнения работы

4.3.1 Открыть архив, созданный в лабораторной работе №3, в пакете программ Microsoft Office Excel. Привести документ к виду, представленному на рис. 4.2.

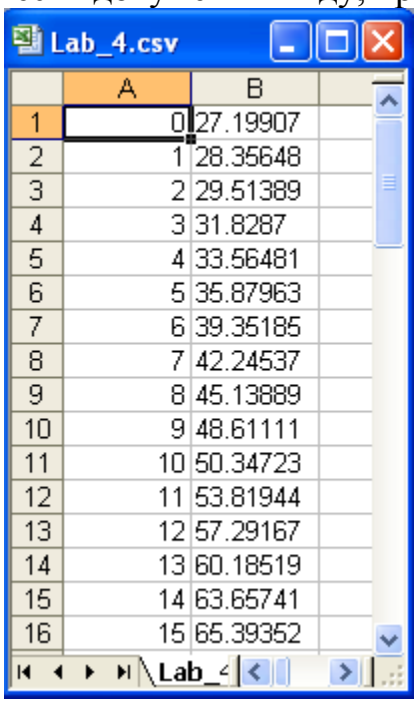


Рис. 4.2 *Документ Microsoft Excel. Зависимость температуры печи от времени*

4.3.2 Открыть изменённый архив в Matlab, выполнив File/Open. В открывшемся окне (см. рис. 4.3) выбрать Next/Finish.

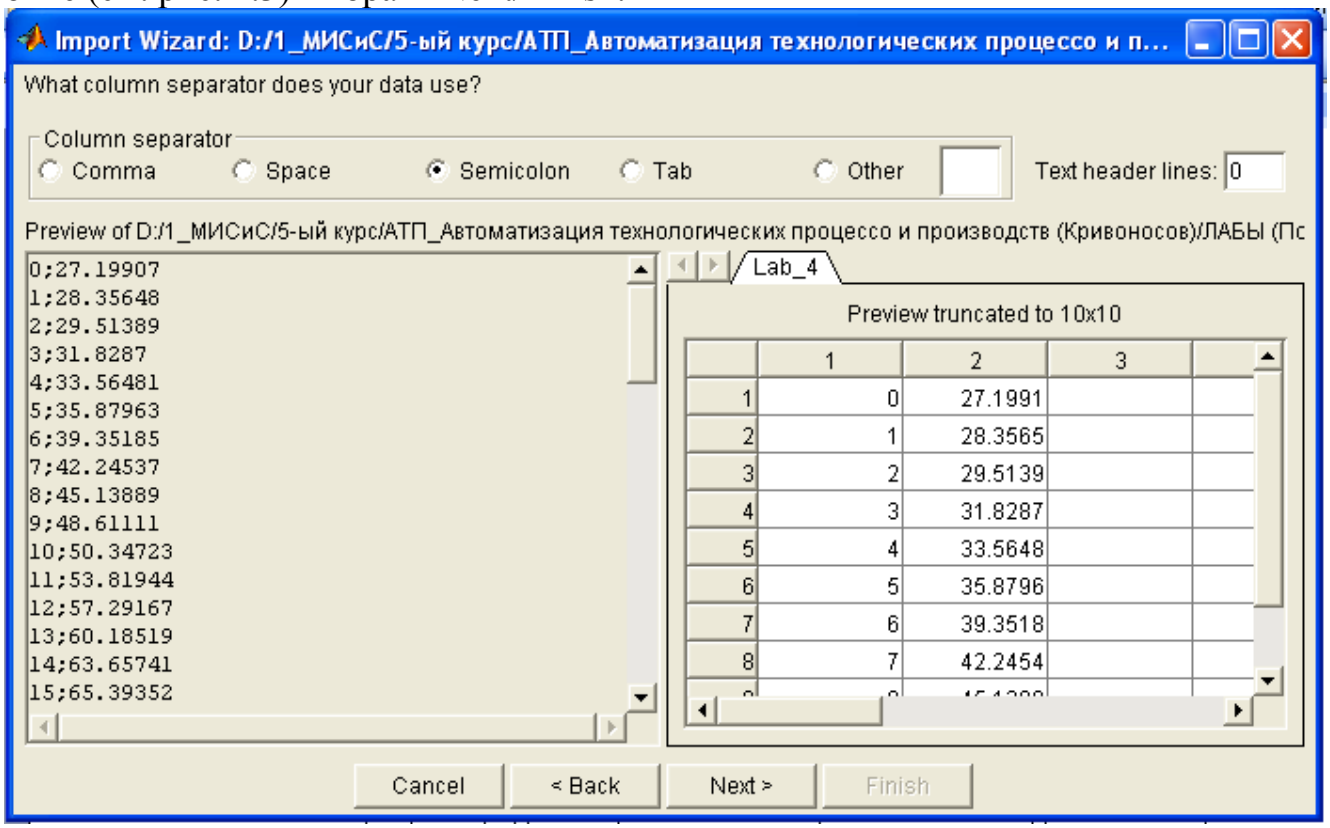


Рис. 4.3 *Зависимость температуры печи от времени в Matlab*

4.3.3 В Matlab запустить Simulink, нажав в панели инструментов на пиктограмму . Создать новый документ, выполнив File/New/Model. Собрать схему, приведённую на рис. 4.4. В блок From Workspace в поле Data вести имя архива “Lab_4”. Запустить проект, нажав на панель инструментов на кнопку Start Simulation . Открыв Scope, проанализировать график переходного процесса (см. рис. 4.5).

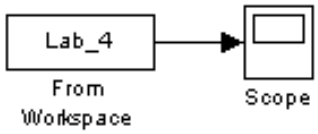


Рис. 4.4 *Схема для построения графика переходного процесса*

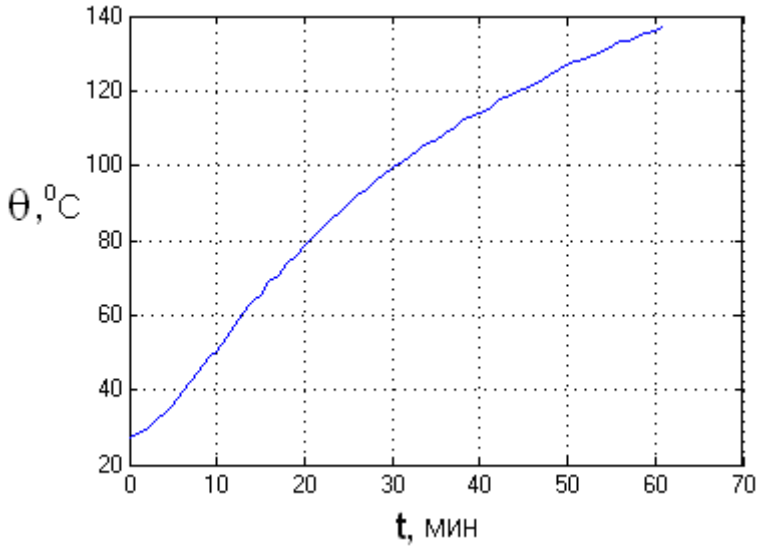


Рис. 4.5 *График переходного процесса*

4.3.4 Для идентификации объекта собрать схему, изображенную на рис. 4.6. Здесь Δx – сигнал управления в %, θ_0 – начальное значение температуры, °C. Для схемы установить дискретность моделируемого времени равную дискретности съема данных при получении переходной характеристики (Вкладка Simulation → Simulation parameters → Solver option → Fixed step + Ode 5 (в соседнем меню)). Далее в Matlab в окне Command Window ввести значения параметров K, T1 и T2 (см. рис. 4.7). После этого открыть блок NCD Outport, выбрать в меню Optimization/Parameteres, в появившемся окне (см. рис. 4.8) ввести оптимизируемые параметры, их нижние и верхние пределы оптимизации. Запустить проект, нажав на панель инструментов на кнопку Start (см. рис. 4.9). Далее в Matlab в окне Command Window просмотреть значения параметров K, T1 и T2 (см. рис. 4.10).

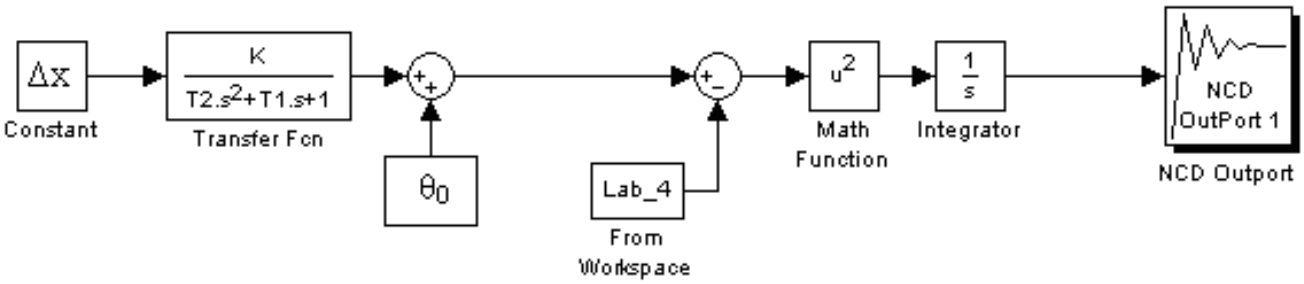


Рис. 4.6 *Схема для идентификации объекта*

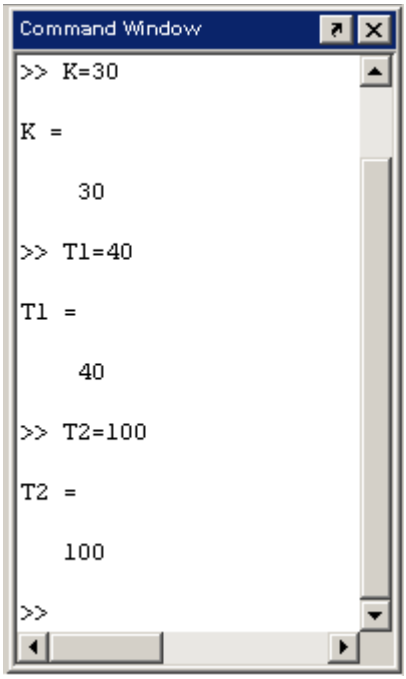


Рис. 4.7 *Ввод параметров K, T1 и T2*

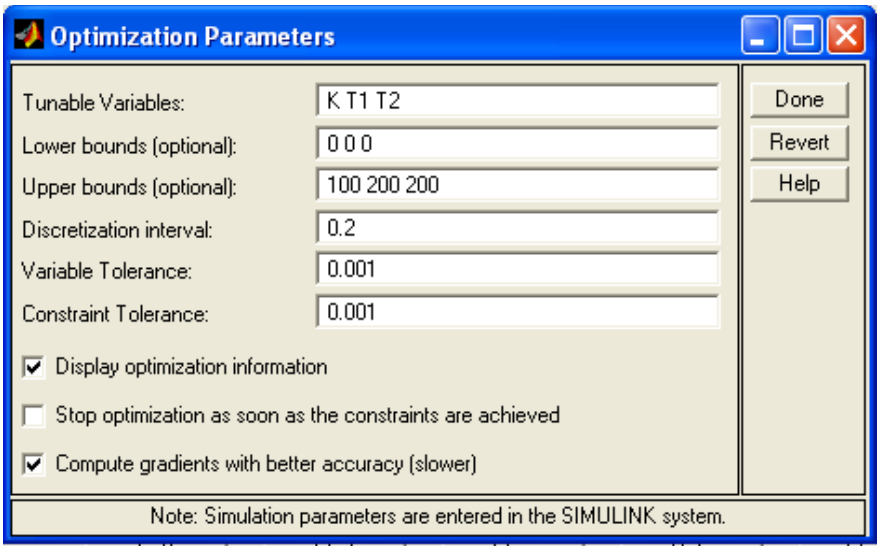


Рис. 4.8 *Определение параметров оптимизации K, T1, T2*

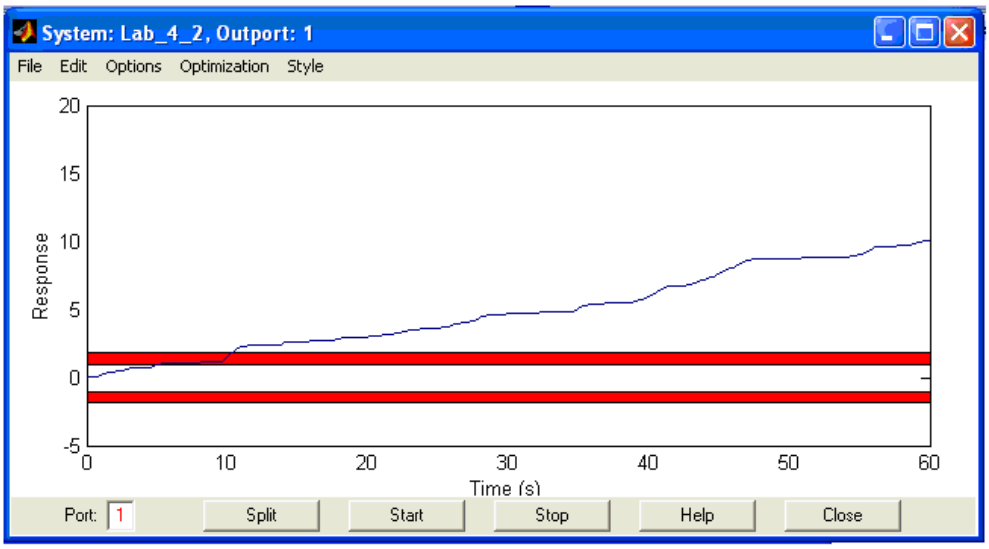


Рис. 4.9 *Результат работы блока NCD Outport*

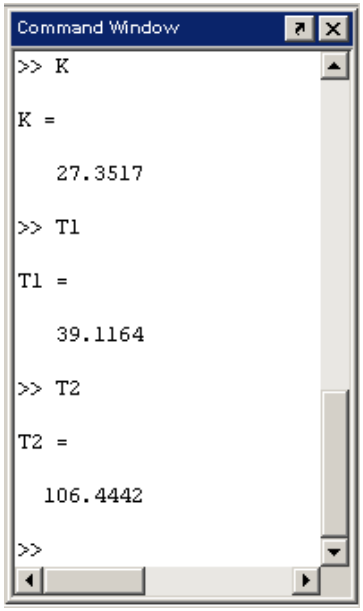


Рис. 4.10 *Просмотр оптимизированных параметров K, T1, T2*

4.3.5 Используя Scope, просмотреть графики переходного процесса, построенных по измеренным (в лабораторной работе №3) и по модельным значениям (см. рис. 4.11). Проанализировать полученные результаты (см. рис. 4.12).

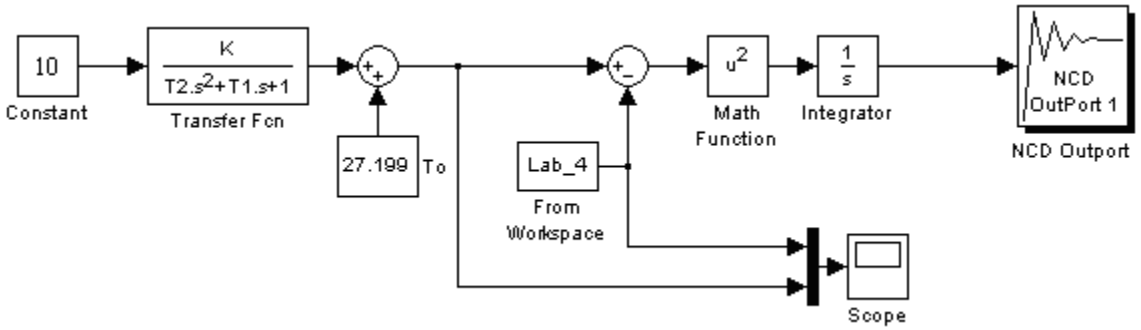


Рис. 4.11 *Схема для просмотра графиков переходного процесса, построенных по измеренным и по модельным значениям*

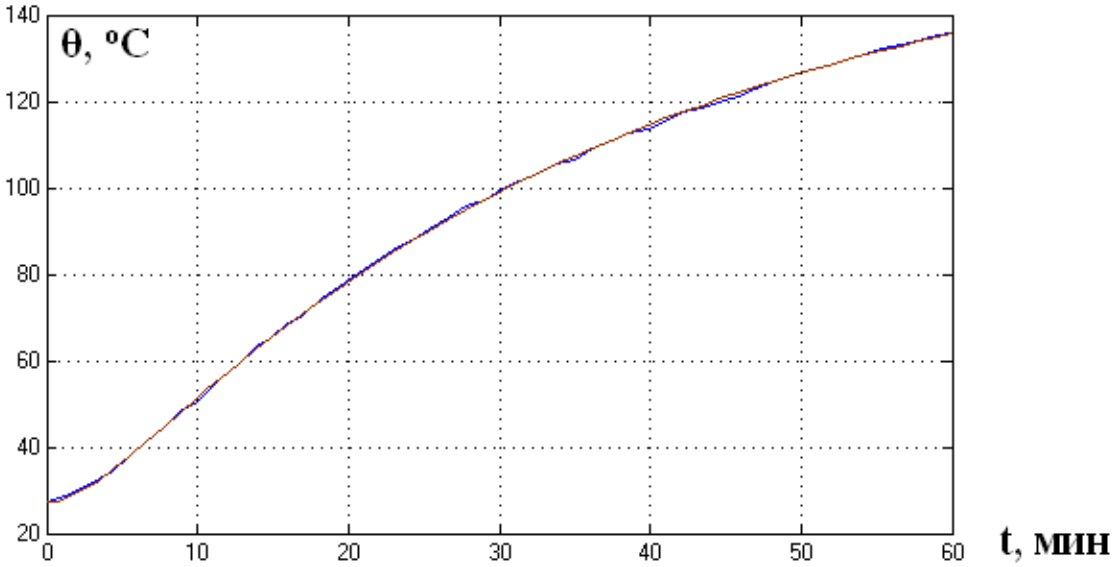


Рис. 4.12 *Графики переходного процесса, построенные по измеренным и по модельным значениям*

4.3.6 Сделать выводы по проделанной работе.

4.4 Контрольные вопросы и задания

4.4.1 Что называется идентификацией объекта?

4.4.2 Каким образом определяется время запаздывания?

4.4.3 Какой вид имеет переходная характеристика апериодического звена 2-го порядка?

4.4.4 Какие величины можно определить по переходной характеристике апериодического звена 2-го порядка?

Обязательные составляющие отчета

1. Переходная характеристика объекта управления.
2. Текст программы или схема оптимизации.
3. График сравнения экспериментальной и модельной переходных функций объекта управления.

Лабораторная работа №5
Определение оптимальных параметров ПИ регулятора

5.1 Цель работы: Ознакомление с особенностями реализации ПИ закона регулирования контроллера. Изучение способов оптимизации коэффициентов регулятора.

5.2 Теоретическое введение

ПИД регулятор контроллера выполняет пропорционально-итегрально-дифференциальное преобразование сигнала и описывается следующей переходной характеристикой:

$$u(t) = K_{\text{у}} \cdot \varepsilon(0) \cdot \left(1 + \frac{1}{\text{T}_{\text{И}}} \cdot t + K_{Д} \cdot e^{\frac{-t}{T_{\text{Д}} / K_{\text{Д}}}} \right),$$

где u(t) – управляющая переменная в автоматическом режиме регулятора;
ε(0) – ступенчатое изменение нормализованного сигнала ошибки;
$К_{\text{У}}$ – усилие регулятора;
$Т_{\text{И}}$ – постоянная времени интегрирования;
$Т_{\text{Д}}$ – время дифференцирования;
$К_{\text{Д}}$ – коэффициент дифференцирования.

Рассматривается задача оптимизации коэффициентов ПИ закона регулирования. Существует много способов решения данной задачи. Например, можно воспользоваться пакетами Siam, Matlab. А также можно применить метод структурного моделирования в комплексе с оптимизацией функционала качества по методу наименьших квадратов.

Оптимизация коэффициентов ПИ регулятора проводится согласно схеме, представленной на рис. 5.1.

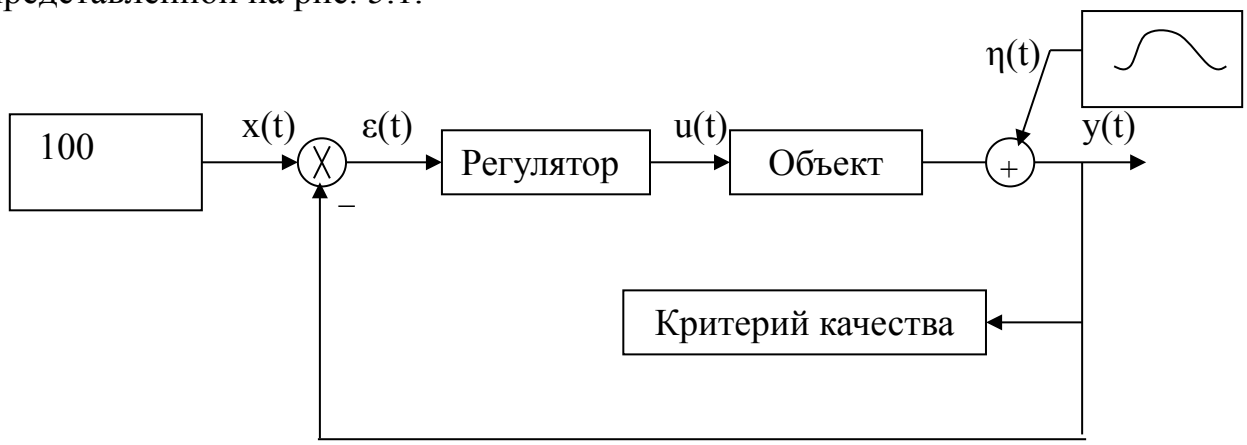


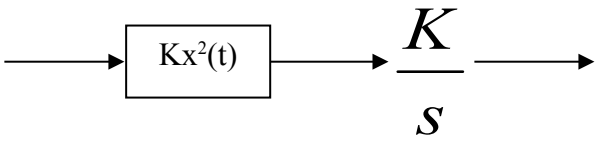
Рис. 5.1 *Схема оптимизации параметров ПИ регулятора*

На рис. 5.1: x(t) – сигнал задания;
ε(t) – сигнал ошибки;
u(t) – сигнал управления;
η(t) – помеха измерения;
y(t) – выход объекта.

При использовании пакета Siam критерий качества выглядит следующим образом:

$$F = \int_0^T \varepsilon(t)^2$$

и имеет структуру для построения в пакете:



При использовании пакета Matlab для оптимизации можно воспользоваться блоком NCD Outport в среде Simulink (см. рис. 5.2). Данный блок позволяет выполнять следующие операции:

1. Задать требуемые ограничения во временной области на любой сигнал оптимизируемой системы;
2. Указать параметры, подлежащие оптимизации;
3. Указать неопределенные параметры;
4. Провести параметрическую оптимизацию системы с учетом заданных ограничений.

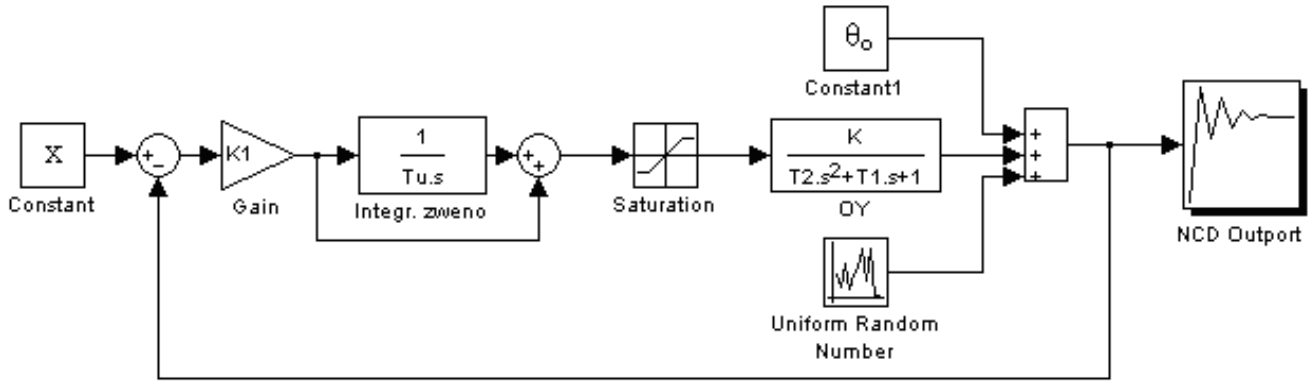


Рис. 5.2 *Пример схемы оптимизации, реализованной в среде пакета Matlab*

5.3 Порядок выполнения работы

5.3.1 Создать схему для оптимизации ПИ коэффициентов одним из предложенных выше способов.

Для этого в Matlab запустить Simulink, нажав в панели инструментов на пиктограмму . Создать новый документ, выполнив File/New/Model. Собрать схему, приведённую на рис. 5.2.

В блоке Saturation ввести значения в поля Upper limit (например, 100) и в Lower limit (например, 0). Допустим, что присутствует шум с равномерным законом распределения в диапазоне [-0.1; 0.1], поэтому в блоке Uniform Random Number в поле Minimum ввести -0.1, а в поле Maximum ввести 0.1.

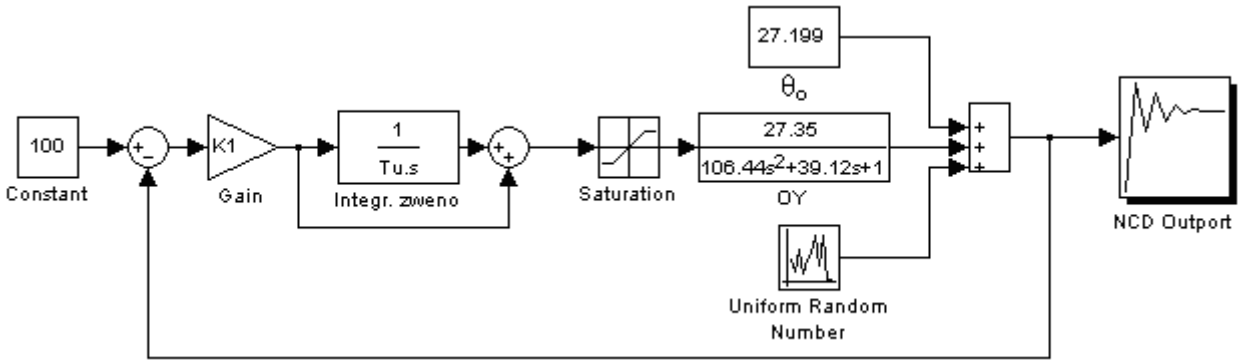


Рис. 5.3 *Схема для оптимизации ПИ коэффициентов*

5.3.2 Провести оптимизацию коэффициентов.

В Matlab в окне Command Window ввести значения параметров K1, Tu (см. рис. 5.4). После этого открыть блок NCD Outport, выбрать в меню Optimization/Parameteres, в появившемся окне (см. рис. 5.5) ввести оптимизируемые коэффициенты, их нижние и верхние пределы оптимизации. Запустить проект, нажав на панель инструментов на кнопку Start (см. рис. 5.6). Далее в Matlab в окне Command Window просмотреть значения параметров K1 и Tu (см. рис. 5.7).

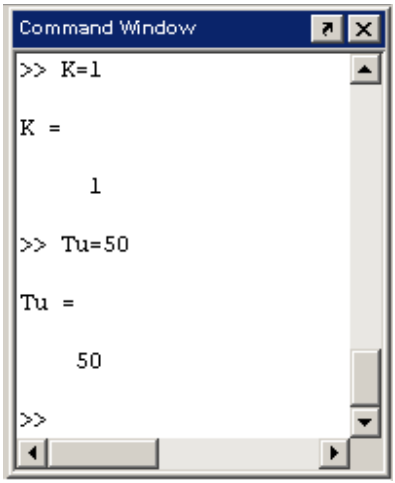


Рис. 5.4 *Ввод параметров K1, Tu*

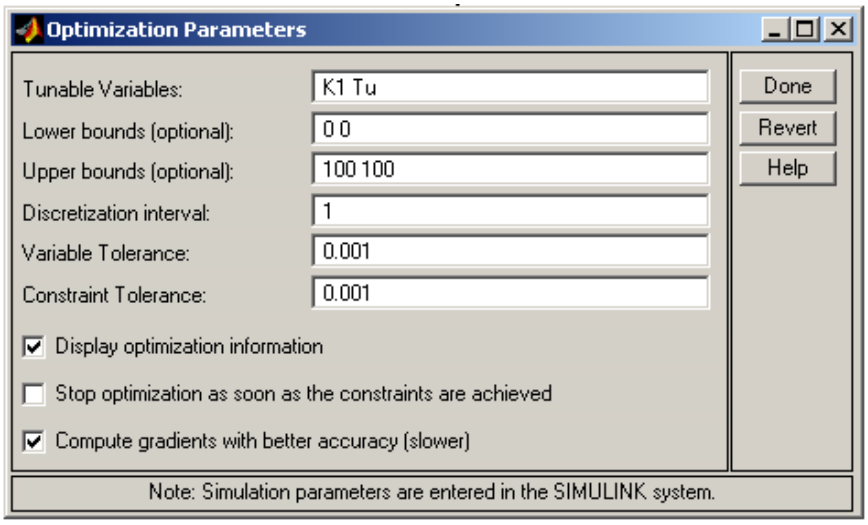


Рис. 5.5 *Определение параметров оптимизации K1, Tu*

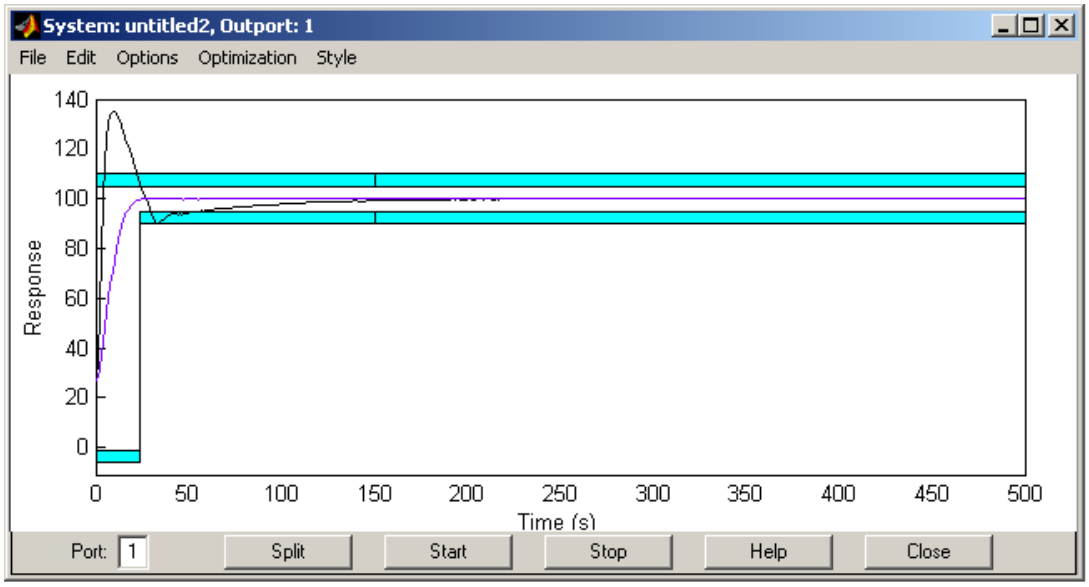


Рис. 5.6 *Результат работы блока NCD Outport*

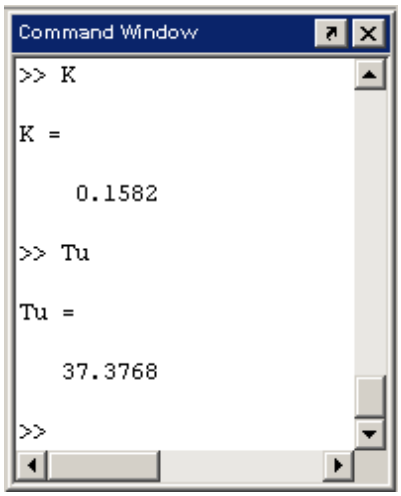


Рис. 5.7 *Просмотр оптимизированных параметров K1 и Tu*

5.3.3 Построить график переходного процесса выхода объекта управления при введенных оптимальных коэффициентах.

Используя Scope, просмотреть графики переходного процесса (см. рис. 5.8). Проанализировать полученные результаты (см. рис. 5.9).

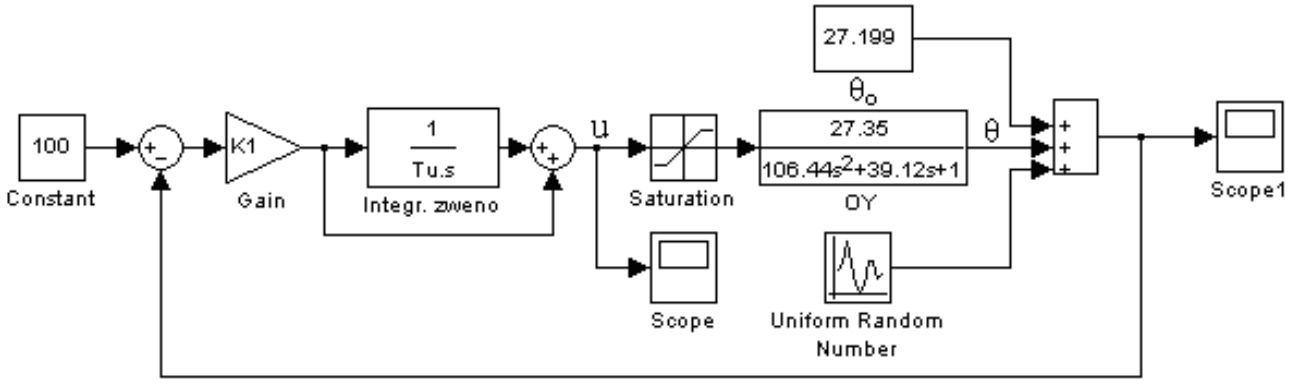


Рис. 5.8 *Схема для просмотра графиков переходных процессов*

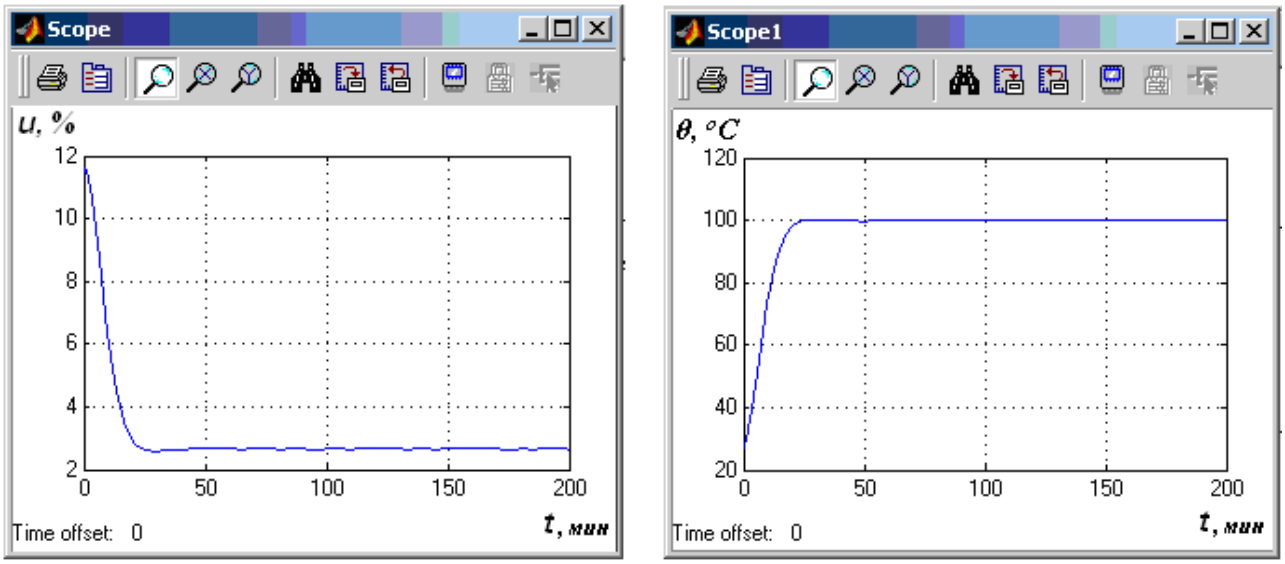


Рис. 5.9 *Графики переходных процессов*

5.3.4 Сделать выводы по проделанной работе.

5.4 Контрольные вопросы и задания

5.4.1 Оцените качество полученного переходного процесса.
5.4.2 Какой метод оптимизации был применен в работе?
5.4.3 Каким образом влияют настроечные коэффициенты ПИД регулятора на качество переходного процесса?
5.4.4 Для чего в схеме модели контура управления объектом (рис. 2) введен блок «saturation»?

Обязательные составляющие отчета

1. Схема оптимизации параметров ПИ регулятора.
2. Значения оптимальных параметров ПИ регулятора.
3. График переходного процесса и управляющего воздействия.

Лабораторная работа №6
Построение системы регулирования температуры

6.1 Цель работы: освоение методики построения типового контура управления с использованием контроллера.

6.2 Теоретическое введение

В АСУ ТП на нижних уровнях иерархии управления находятся подсистемы автоматического управления и регулирования технологическими агрегатами и отдельными параметрами. Функциональная структура типового контура автоматического регулирования приведена на рис. 6.1.

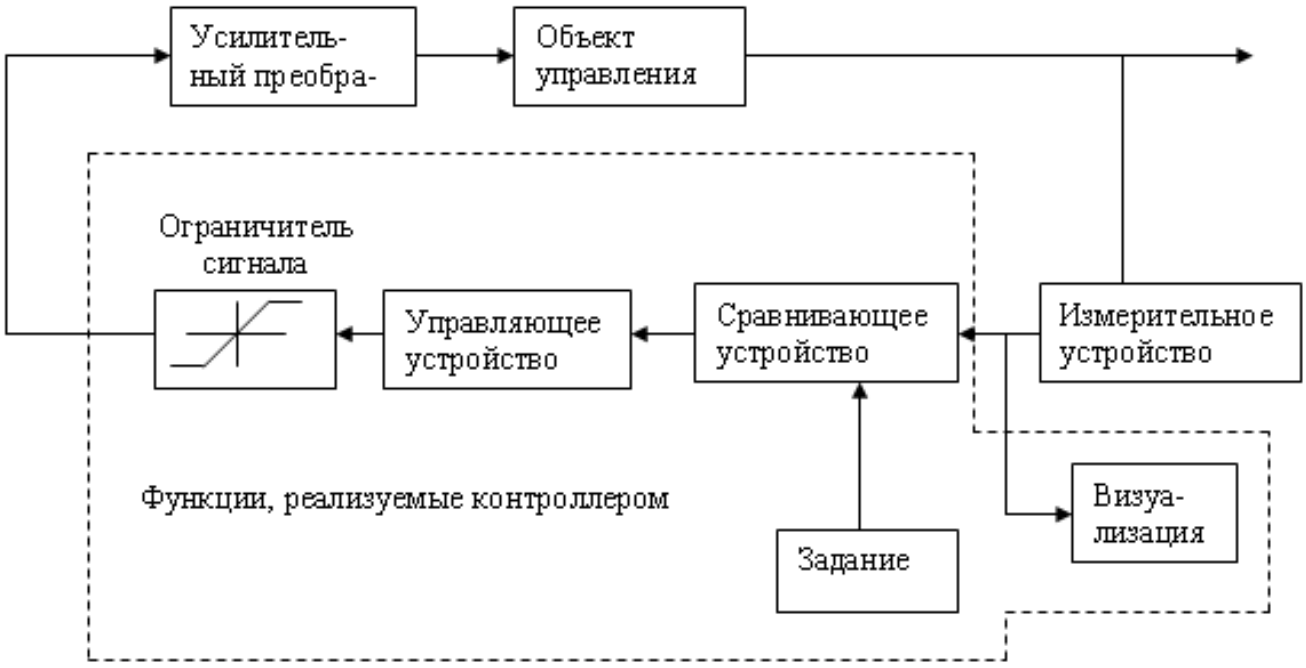


Рис. 6.1 *Функциональная схема типового контура регулирования*

В данной лабораторной работе рассматривается контур управления температурой в электрической нагревательной печи. Объектом управления является печь, выходная величина которой – температура – измеряется при помощи термопары или термосопротивления.

Для стабилизации значения температуры в STEP 7 при построении контура управления на базе ПИД, ПИ регуляторов рекомендуется использовать блок (TCONT_CP), который реализует функции указанных регуляторов.

Вызов блока регулятора

Схема, изображенная на рис. 6.2, показывает вызов управления в FBD. А в таблице 1 приведено описание некоторых параметров блока ПИД-регулирования (TCONT_CP).

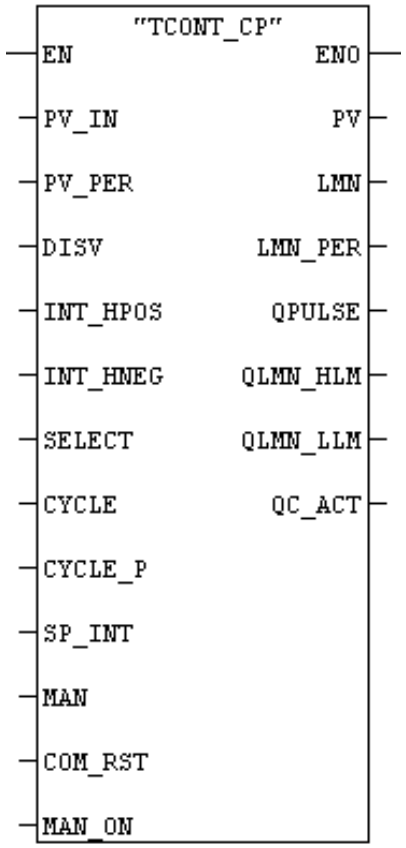


Рис. 6.2 *Схема TCONT_CP*

Таблица 1. Описание используемых в работе параметров блока TCONT_CP

Параметр	***Назначение***	***Тип***	***Описание***
PV_IN	INPUT (Вход)	REAL	Вход переменной процесса (“Process Variable In”). Начальное значение может быть установлено на данном входе или может быть подключена внешняя переменная процесса в формате чисел с плавающей запятой.
SP_INT	INPUT (Вход)/ OUTPUT (Выход)	REAL	Внутреннее значение сигнала уставки (“Internal Setpoint”). Вход “Internal Setpoint” используется для задания уровня сигнала уставки.
MAN	INPUT (Вход)/ OUTPUT (Выход)	REAL	Управляющая переменная, введенная вручную (“Manual Value”). При автоматическом режиме корректируется до значения управляющей переменной.
MAN_ON	INPUT (Вход)/ OUTPUT (Выход)	BOOL	Переключатель на работу в ручном режиме (“Manual Operation On”). Если параметр “Manual Operation On” установлен, то управляющая переменная, введенная вручную (MAN),

			устанавливается в качестве значения управляющей переменной. 0 – автоматический режим; 1 – ручной режим.
LMN	OUTPUT (Выход)	REAL	Управляющая переменная (“Manipulated Variable”). Действующее значение управляющей переменной в формате числа с плавающей запятой подается на одноименный выход: Manipulated Variable.
QPULSE	OUTPUT (Выход)	BOOL	Выходной импульсный сигнал (“Output Pulse”). ШИМ-модулированное импульсное представление управляющей переменной на выходе: Output Pulse.

Порядок конфигурирования блока, реализующего ПИ-закон регулирования:

1) Создать организационный блок OB35; открыть его и из справочника Overviews взять компонент FB58 TCONT_CP CONTROL по ветви Libraries / Standard Library / PID Control Blocks (см. рис. 6.3).

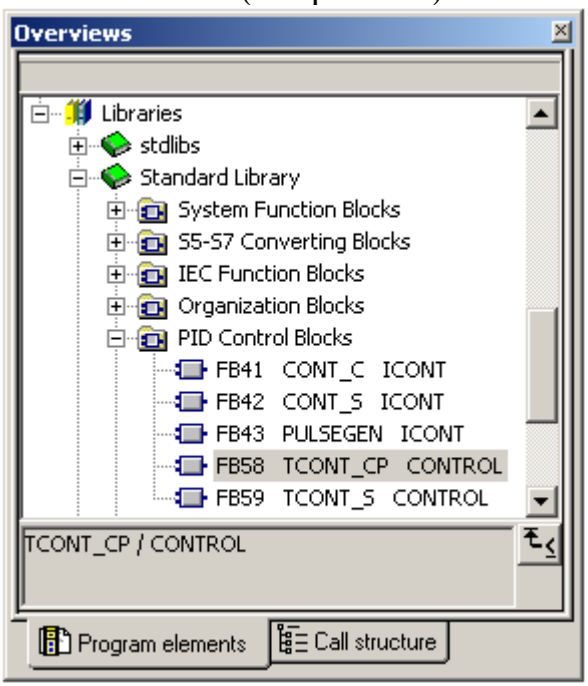


Рис. 6.3 *Выбор блока, реализующего ПИ-закон регулирования, из справочника Overviews*

2) Необходимо выделить место в памяти контроллера для ПИ-регулятора. Для этого надо создать блок данных DB58.

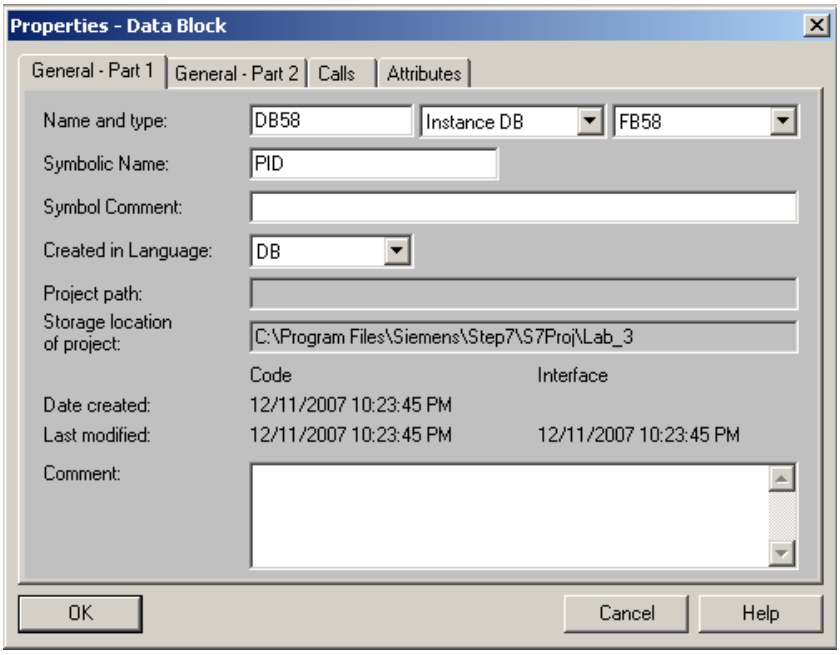


Рис. 6.4 *Окно параметров блока данных DB58*

3) После того как создали DB58, в OB35 в заголовке блока TCONT_CP выбрать “PID” (см. рис. 6.5).

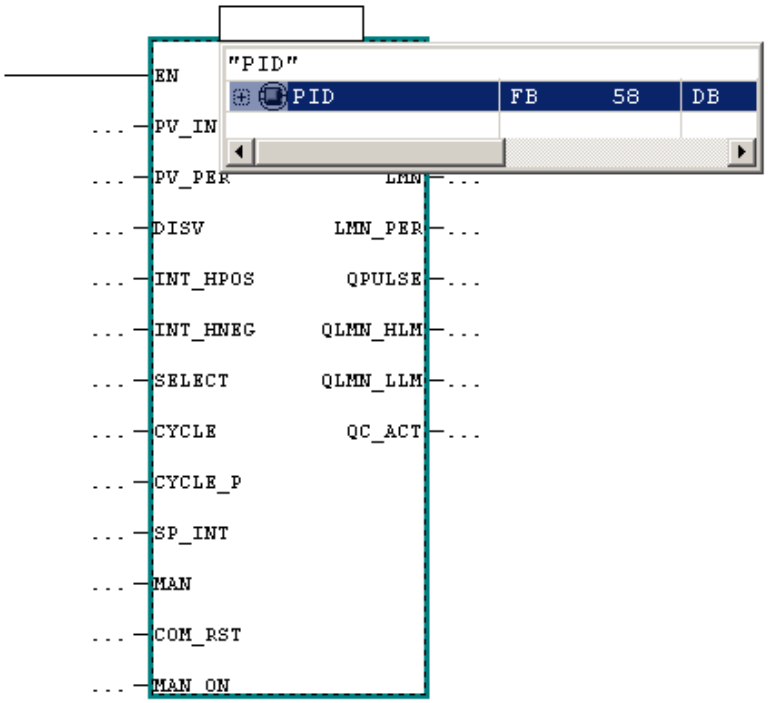


Рис. 6.5 *Привязка блока ПИ-регулирования (TCONT_CP) к DB58*

4) Открыть блок DB58 и ввести значения требуемых параметров (рис. 6.6).

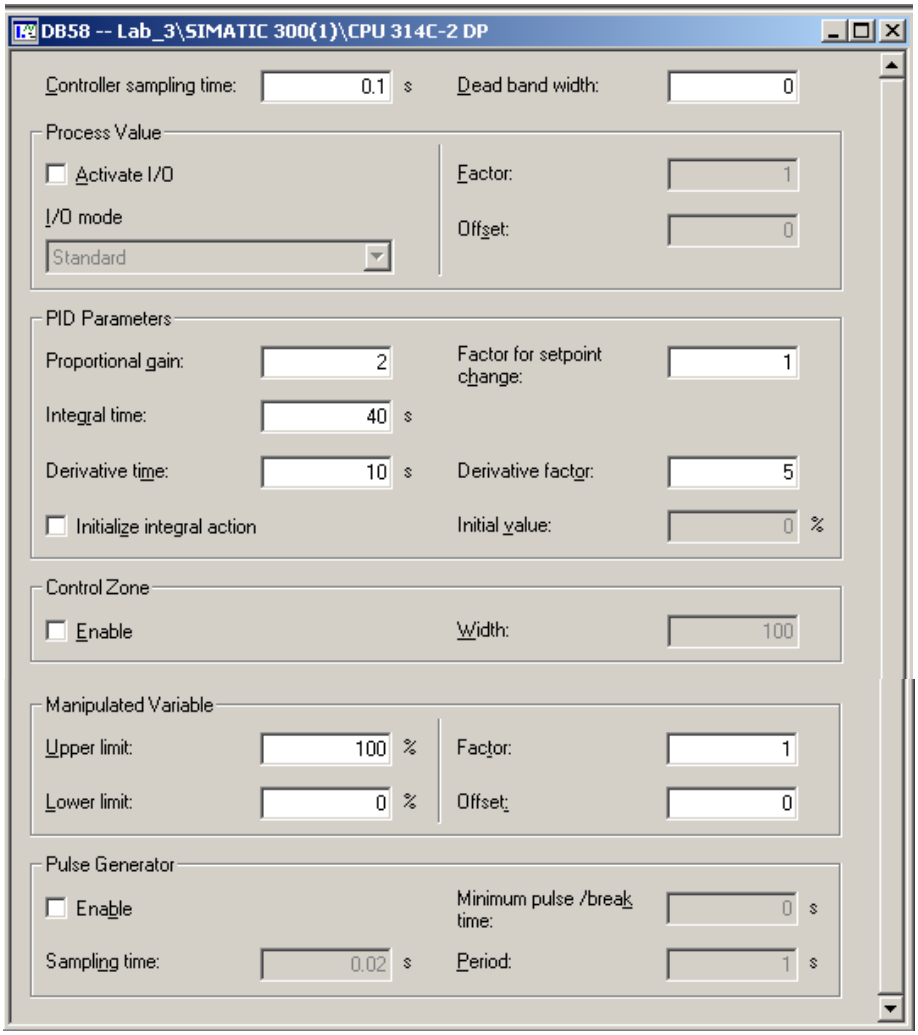


Рис. 6.6 *Окно ввода значений параметров блока данных DB58*

6.3 Порядок выполнения работы

6.3.1 Создать новый проект и сконфигурировать контроллер в STEP7.

6.3.2 В левой части окна проекта раскрыть дерево до Blocks. Создать организационный блок OB35. В появившемся окне Properties выбрать язык, на котором будем писать программу: LAD (контактно-релейные схемы).

6.3.3 В появившемся окне написать программу выполнения поставленной задачи (реализовать систему регулирования температуры) (см. рис. 6.7):

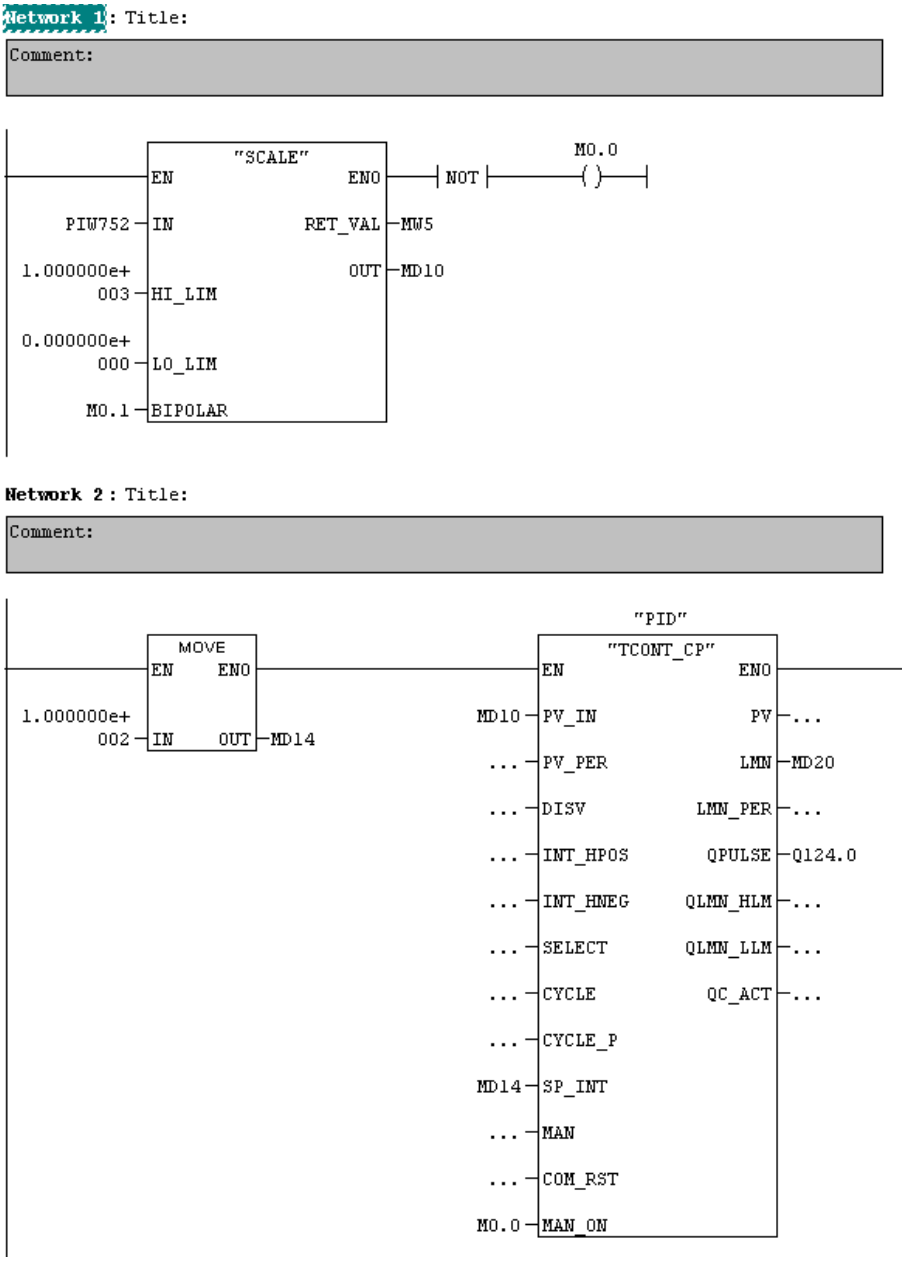


***Рис. 6.7** Программа выполнения поставленной задачи*

6.3.4 Сохранить изменения в OB35, выбрав в главном меню File/Save.

6.3.5 Открыть блок данных DB58, в который ввести значения коэффициента пропорциональности $K_П$ и время интегрирования $T_И$ (перевести в секунды), полученные в лабораторной работе №5. А также установить Period равным значению при реализации широтно-импульсной модуляции в лабораторной работе №3. Параметр Sampling time установить в 0,1с. Установить флаг Pulse Generator (см. рис. 6.8).

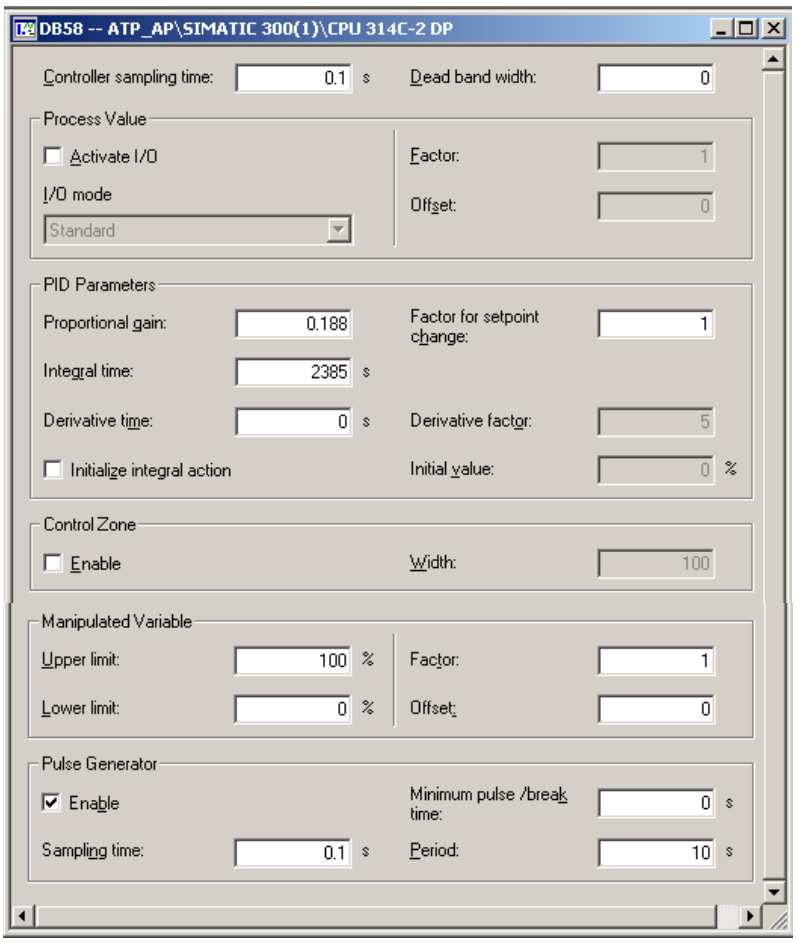
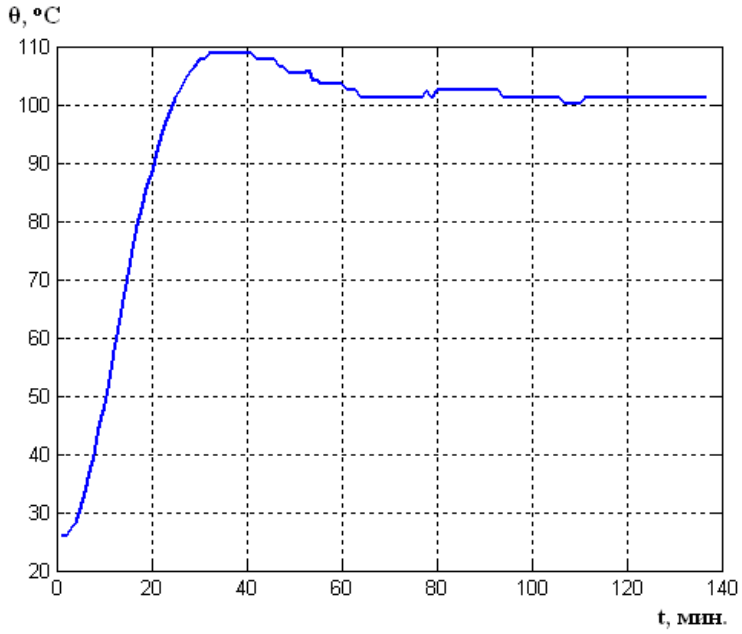


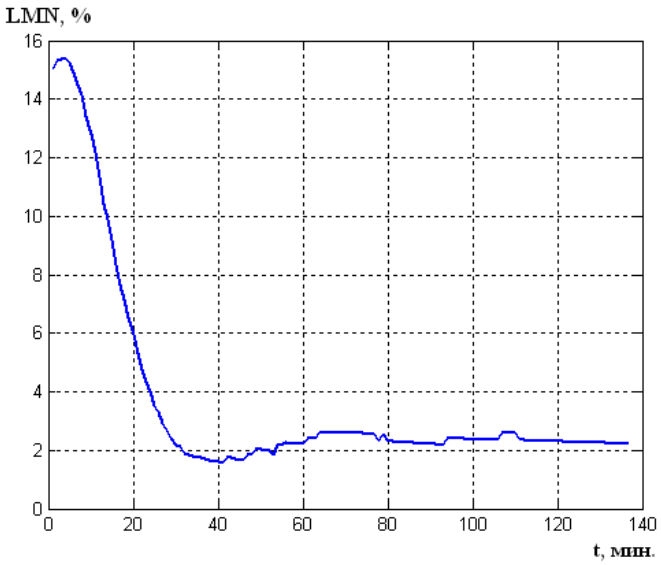
Рис. 6.8 *Ввод значений $K_П$ и $T_И$ в блок DB58*

6.3.6 Реализовать визуальное отображение и архивирование сигналов температуры и управления (LMN) средствами ProTool.

6.3.7 Построить график кривой переходного процесса (см. рис. 6.9). Это можно реализовать с помощью средств пакетов прикладных программ Microsoft Excel или Matlab.



а)



б)

Рис. 6.9 *Графики переходных процессов, построенные по значениям: а) тега VAR_1; б) тега VAR_2*

6.3.8 Сделать выводы по проделанной работе.

6.4 Контрольные вопросы и задания

6.4.1 Что называется перерегулированием и как оно вычисляется?
6.4.2 Поясните роль алгоритмов в схеме, приведенной на рисунке 6.7.
6.4.3 Поясните принцип функционирования каналов ПИ регулятора.
6.4.4 Почему в Step 7 программу регулирования рекомендуется писать в OB35, а не в OB1?

Обязательные составляющие отчета

1. Программа на STEP 7.
2. График переходного процесса.
3. Значения показателей качества переходного процесса.

Список литературы

1. **Лазарев, Ю.** Моделирование процессов и систем в MATLAB. [Текст]: Учебный курс. / Ю. Лазарев. - Спб.: Питер; Киев: Издательская группа BHV, 2005, 512с.: ил.; 23 см.; 3000 экз. – ISBN 5-469-00600-X

2. Методы классической и современной теории автоматического управления: Учебник в 3-х т. - Т.3: Методы современной теории автоматического управления / Под ред. Н.Д. Егупова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2000.-748 с., ил. 22 см.

3. Методы классической и современной теории автоматического управления [Текст]. В 5-и т. Т. 2. Статистическая динамика и идентификация систем автоматического управления / Под ред. К.А. Пупкова, Н.Д. Егупова. – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. - 640 с.: ил.; 24,5 см. – Библиогр. с. 624-631.; предм. указ.: с. 619-623. – 2500 экз. – ISBN 5-7038-2190-8 (Т.2)

4. **Ганс, Бергер.** Автоматизация с помощью Программ STEP7 LAD и FBD Программируемые контроллеры SIMATIC S7-300/400 Заказной номер: 6ES7810-4CA05-8AR0. Издание 2-е переработанное, 2001SIMATIC HMI. ProTool: Конфигурирование систем на базе Windows. Руководство пользователя. Заказной номер 6AV6594-1MA05-2AB0. Версия от 12/99.

Учебное издание

Полещенко Дмитрий Александрович
Кривоносов Владимир Алексеевич

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ И ПРОИЗВОДСТВ

Методические указания
к выполнению лабораторных работ

Редактор: Иванова Н.И.
Компьютерный набор: Полещенко Д.А.
Корректор: Иванова Н.И.

Подписано в печать_____ Бумага для множительной техники

Формат_____ Усл. печ. л._____ Тираж_____экз. Заказ_____

Отпечатано с авторского оригинала
в отделе оперативной печати СТИ МИСиС
г. Старый Оскол, м-н Макаренко 40