ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ

Библиотека ARM7_specific

Руководство по применению

Версия 01

Москва

2010

Содержание

Введение

Компания ОВЕН предоставляет пользователю библиотеки дополнительных программных компонентов, облегчающие составление проекта работы программируемого логического контроллера (ПЛК) для решения наиболее распространенных практических задач. Приводимые в данном описании библиотеки предназначены для работы на контроллерах ОВЕН с встроенными жидкокристаллическими индикаторами (ЖКИ) для управления отображением информации на экране ЖКИ.

Библиотеки поставляются в виде файлов на компакт-диске, входящем в комплект поставки ОВЕН ПЛК (папка «Lib\Библиотеки ОВЕН»).

Библиотека ARM7_specific (файл ARM7_specific.lib) используется для работы ПЛК с отображением информации на экране ЖКИ.

Назначение всех компонентов библиотек указано в таблице 1.

Таблица 1

Имя компонента	Назначение и область применения
GetOutputType	Выдача информации о типе выходного устройства, номер которого передан функции
READ_BY_HASH	Получение дескриптора параметра
SAVE_PAR_ATTRIB	Сохранение атрибутов доступа параметра
SAVE_PARAMETER	Сохранение значения параметра

Внимание! У программных компонентов библиотек режим симуляции (Simulation Mode) не предусмотрен. Отладка программы проводится при подключенном контроллере, – программные компоненты при этом работают только в самом контроллере.

Словарь условных сокращений и терминов

Далее в тексте для компактного описания используются следующие сокращения:

CoDeSys – Controllers Development System, программное обеспечение, специализированная среда программирования логических контроллеров. Торговая марка компании 3S-Smart Software Solutions GmbH.

ST – Structured Text, структурированный текст, язык программирования по МЭК 61131-3.

ЖКИ – жидкокристаллический индикатор, расположенный на лицевой панели ПЛК.

ПЛК – программируемый логический контроллер.

РП – руководство пользователя на программируемый логический контроллер (поставляется на компакт-диске вместе с контроллером).

0 и **1** – при описании переменных типа BOOL нулю соответствует значение «FALSE»; единице – значение «TRUE».

В тексте используются следующие технические термины:

Экран – пронумерованная совокупность данных, программно заданных пользователем для отображения на ЖКИ. Число доступных для вызова на ЖКИ экранов (рабочих экранов) задается функцией «setworkscreencount», либо (если указанная функция не вызывалась) задается в параметре «Quantity of work screens» модуля «WorkInd» в разделе «Конфигурация ПЛК (PLC Configuration)», см. РП. Максимально в режиме индикации может быть доступно до 16 экранов (нумеруемых от 0 до 15), каждый из которых может быть выведен на ЖКИ.

Рабочий режим индикации (work_mode) – режим, при котором пользователь может управлять выбором отображаемого на ЖКИ экрана и работой ПЛК при помощи кнопок на лицевой панели.

Дескриптор – структура, однозначно описывающая выбранный параметр дерева «Конфигурация ПЛК (PLC_configuration)»: значение, минимум, максимум, адрес modbus и т. д. (см. РП).

Тип данных – формат представления переменных согласно РП.

1. Установка дополнительных библиотек для проекта

В CoDeSys все файлы библиотек дополнительных программных компонентов имеют расширения *.lib (Library) и находятся в папке Library. Она расположена по месту размещения основной программы на диске компьютера (по умолчанию – C:\Program Files\3S Software\CoDeSys V2.3\Library).

По умолчанию подключен (доступен) стандартный набор библиотек. Дополнительные библиотеки добавляются пользователем по мере необходимости в папку к уже имеющимся библиотекам. Для подключения новых библиотек к проекту соответствующие файлы переписываются пользователем в ту же папку, где находятся все используемые библиотеки.

Чтобы увидеть, какие библиотеки уже подключены к проекту, и подключить дополнительные библиотеки, используется «Менеджер библиотек (Library Manager)», – его можно открыть из главного меню CoDeSys командами «Окно (Window) ▸ Менеджер библиотек (Library Manager)» или выбором на вкладке организатора объектов «Ресурсы (Resources)» режима работы «Менеджер библиотек (Library Manager)», см. рисунок 1.1. В средней верхней части появившегося окна отображается список установленных библиотек.

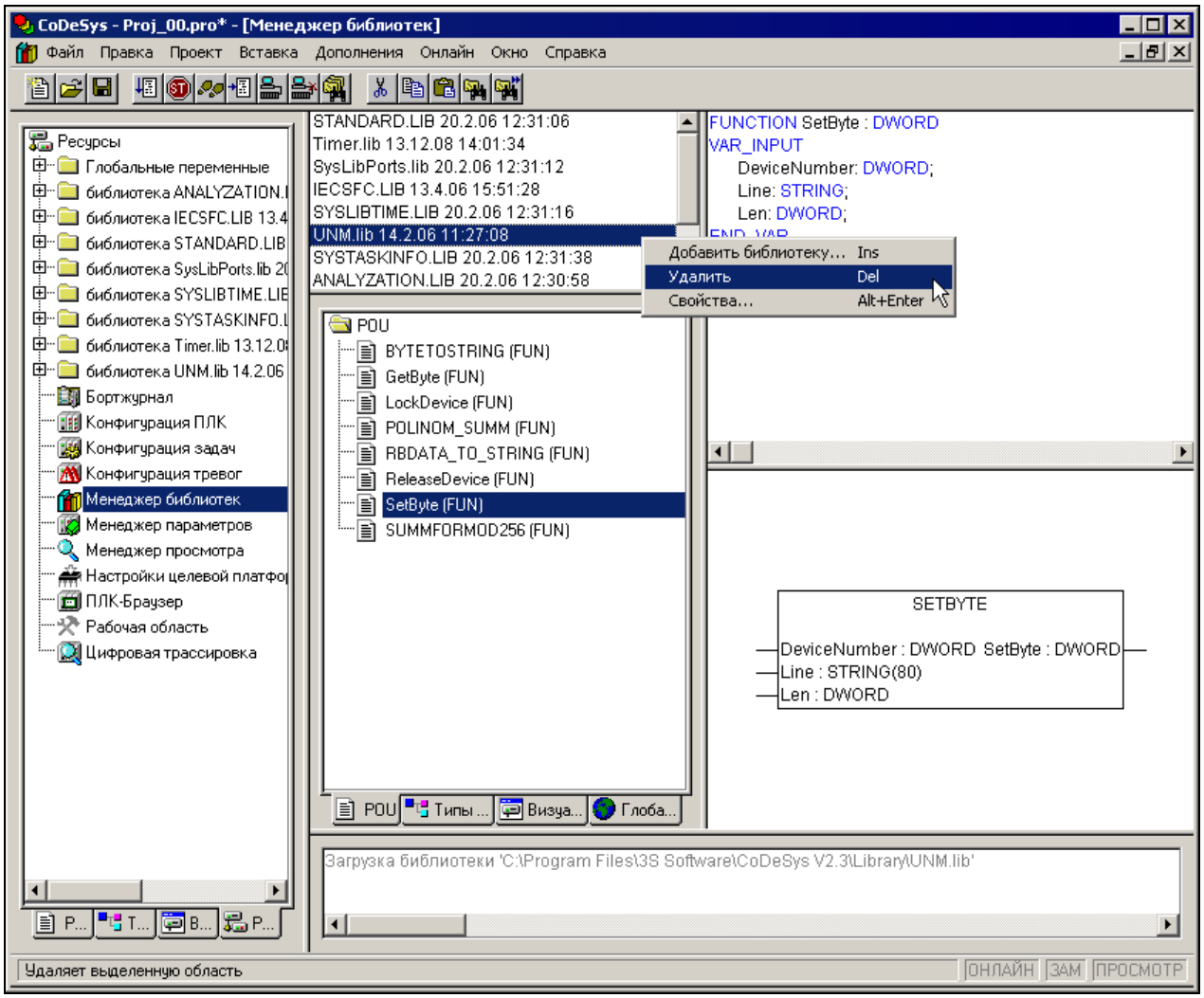


Рисунок 1.1 – Окно вкладки организатора объектов «Ресурсы (Resources)» с режимом работы «Менеджер библиотек (Library Manager)»

Установка дополнительных библиотек выполняется из главного меню последовательным выбором команд: **Вставка (Insert)** ▸ **Добавить библиотеку (Additional Library)** ▸ в открывшемся окне папки Library (рисунок 1.2) выделяется файл с именем нужной библиотеки (например, ARM7_specific.lib) и дается команда **Открыть**.

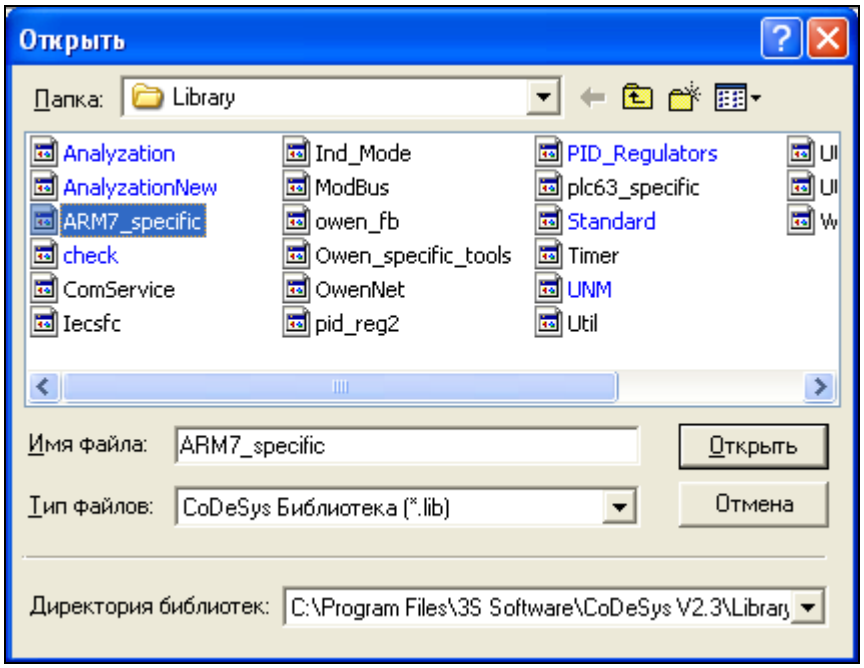


Рисунок 1.2 – Окно выбора подключаемой к проекту дополнительной библиотеки

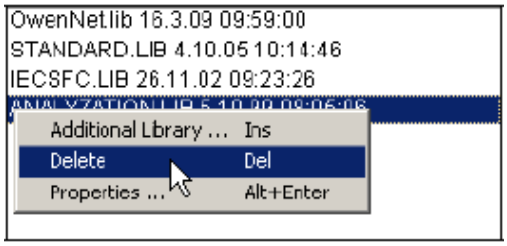
Теперь в перечне библиотек, доступных в проекте, появится вновь установленная библиотека.

Для просмотра состава и свойств программных компонентов курсором выбирается нужная библиотека, – при этом появится папка с программными компонентами, в которой выделяется конкретный программный компонент (на рисунке 1.1 справа дана краткая справочная информация по его использованию).

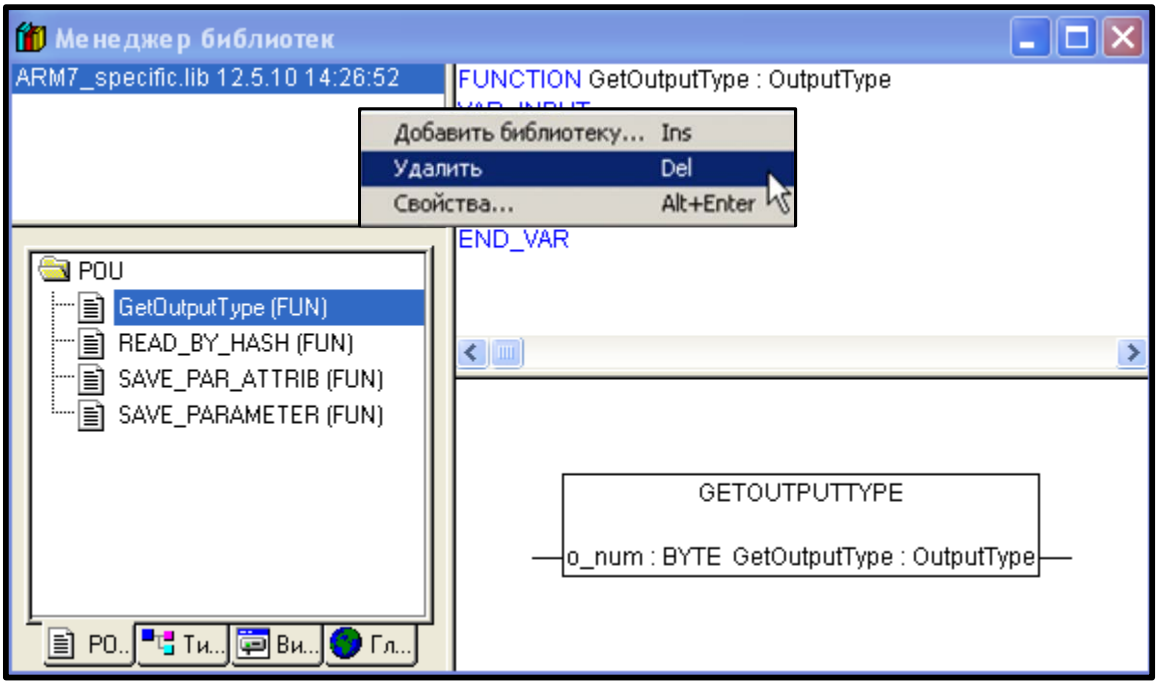
Примечания.

1. Рекомендуется размещать все библиотеки, которые планируется подключать, в папке для хранения библиотек, создаваемой CoDeSys автоматически.
2. Для каждого нового проекта добавление новых библиотек проводится индивидуально, при необходимости их применения.

Удаление выделенной библиотеки выполняется из контекстного меню командой **Удалить (Delete)** (или из главного меню командой **Правка (Edit)** ▸ **Удалить (Delete)**, рисунок 1.3 (или нажатием клавиши <Delete>).



а)



в)

Рисунок 1.3 – Удаление дополнительной библиотеки:
а) для CoDeSys с английским интерфейсом; в) для CoDeSys с русским интерфейсом

2. Состав библиотеки ARM7_specific

Библиотека предназначена для переключения между режимами индикации ЖКИ и для вывода строковых констант на ЖКИ в монопольном рабочем режиме. Библиотеку следует использовать при создании нестандартных способов отображения сообщений на ЖКИ, когда функций экранного рабочего режима не хватает.

Библиотека использует структуры данных, показанные на рисунке 2.1.

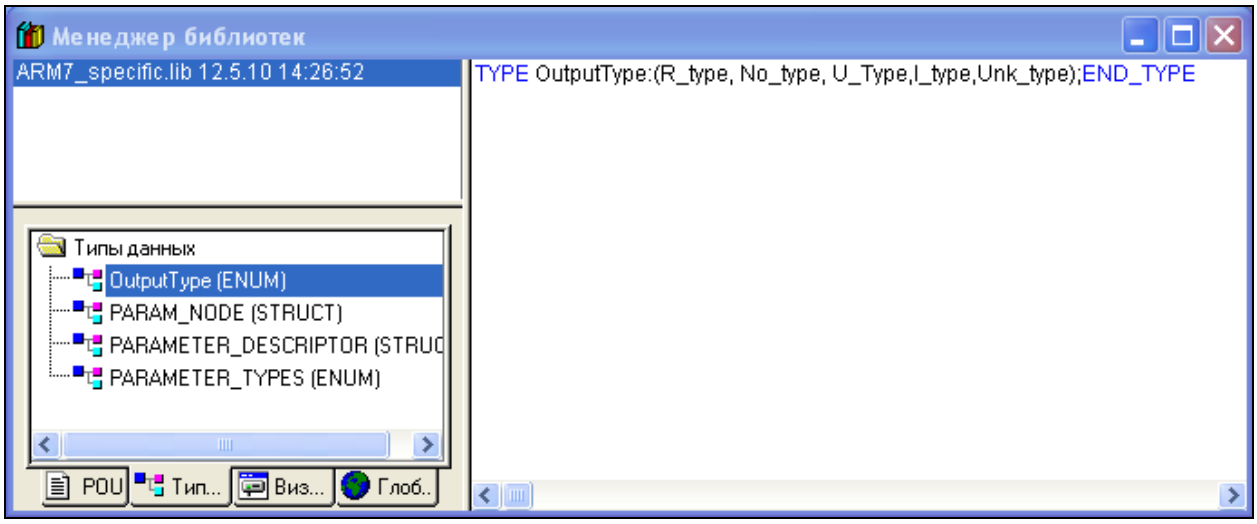


Рисунок 2.1 – Окно «Менеджер библиотек (Library Manager)», отображающее пользовательские типы данных

Внимание! Структуры пользовательских типов данных редактировать нельзя.

Выдача информации о типе выходного устройства (GetOutputType)

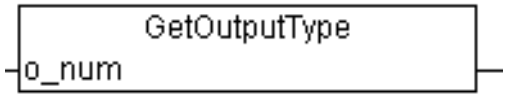


Рисунок 2.2 – Структурная схема

Таблица 2.1

Имя программного компонента	GetOutputType		
Тип программного компонента	Функциональный блок ☐	Функция ☒	Программа ☐
Особенности работы	Для работы не требуется установка в проекте дополнительных библиотек		
Применение на контроллерах	ПЛК63, ПЛК73, ПЛК410		

Входная переменная:	**Тип данных**	**Пояснения**
o_num	BYTE	Номер выхода (нумерация начинается с 1)

Выходная переменная:	**Тип данных**	**Пояснения**
GetOutputType	OutputType	Возвращаемое значение – тип выхода (см. таблицу 2.2)

Описание работы

Функция возвращает значение типа выхода ПЛК согласно таблице 2.2.

Таблица 2.2

Целочисленное значение	**Строковый псевдоним***	**Описание**
0	R_type	Дискретный выход («Р», «К», «С» или «Т»)
1	No_type	Ошибка конфигурации
2	U_type	Выход типа «У»
3	I_type	Выход типа «И»
4	Unk_type	Ошибка конфигурации
* Строковое условное обозначение целочисленного значения типа данных.		

Пример вызова данной функции в составе программы, которая на экране дисплея отображает тип соответствующего выхода, на языке ST выглядит следующим образом:

```
IF GetOutputType(2) THEN ShowString(i-1,0,1,'Дискретный'); END_IF;
```

Указанный пример выведет на дисплее надпись «Дискретный» при условии, что тип выхода 2 – дискретный.

Чтение дескриптора параметра (READ_BY_HASH)

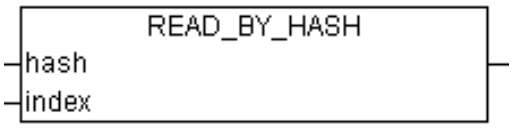


Рисунок 2.3 – Структурная схема

Таблица 2.3

Имя программного компонента	READ_BY_HASH		
Тип программного компонента	Функциональный блок ☐	Функция ☒	Программа ☐
Особенности работы	Для работы не требуется установка в проекте дополнительных библиотек		
Применение на контроллерах	ПЛК63, ПЛК73, ПЛК410		

Входные переменные:	**Тип данных**	**Пояснения**
hash	WORD	Хэш параметра по протоколу ОВЕН (задается пользователем)
index	WORD	Линейный индекс параметра по протоколу ОВЕН. Диапазон значений от 0 до 65534 (задается пользователем)

Выходная переменная:	**Тип данных**	**Пояснения**
READ_BY_HASH	POINTER TO PARAMETER_DESCRIPTOR	Указатель на дескриптор параметра

Описание работы

Функция возвращает значение дескриптора параметра. Основное практическое назначение чтения дескриптора – независимость программного кода от значений, заданных в параметрах дерева «Конфигурация ПЛК (PLC_configuration)», см. РП.

Например, требуется на ЖКИ отобразить минимальное значение параметра **tst1** (хэш-код **0x9C17**) с индексом 1 – для этого вводим дополнительную переменную **a** типа **pointer to parameter_descriptor.**

Программный код на языке ST будет выглядеть следующим образом:

```
a:=Read_By_Hash(16#9C17,1);
ShowReal(0,0,0,'%2.1f',a^.low_diap);
```

Сохранение атрибутов доступа параметра (SAVE_PAR_ATTRIB)

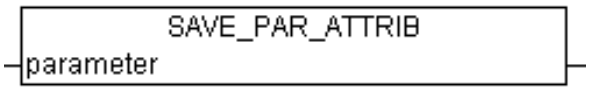


Рисунок 2.4 – Структурная схема

Таблица 2.4

Имя программного компонента	SAVE_PAR_ATTRIB		
Тип программного компонента	Функциональный блок ☐	Функция ☒	Программа ☐
Особенности работы	Для работы не требуется установка в проекте дополнительных библиотек		
Применение на контроллерах	ПЛК63, ПЛК73, ПЛК410		

Входные переменные:	**Тип данных**	**Пояснения**
parameter	POINTER TO PARAMETER_DESCRIPTOR	Указатель на параметр

Выходная переменная:	**Тип данных**	**Пояснения**
Save_par_attrib	INT	Функция ничего не возвращает

Описание работы

Функция записывает атрибуты доступа параметра:

parameter^.flags.0 – атрибуты доступа с панели (аналогично галке «разрешить изменение с передней панели» в параметрах дерева «Конфигурация ПЛК (PLC_configuration)», см. РП;

parameter^.flags.1 – атрибуты доступа по сети (аналогично галке «разрешить изменение по сети» в «Конфигурация ПЛК (PLC_configuration)»).

Принимаемые значения: 0 – разрешить изменение, 1 – запретить изменение.

Внимание! Все остальные значения в записываемой структуре parameter_descriptor должны полностью соответствовать редактируемому параметру.

Основное назначение изменения атрибутов из CodeSys – реализация двух основных режимов функционирования программы: рабочего (work_mode) и конфигурационного. В рабочем режиме осуществляется блокировка ряда параметров с целью запрета их случайного изменения пользователем.

Применение данной функции показано на примере ее вызова в составе программы, которая имеет два режима: **run** и **stop**. Переход из режима **stop** в режим **run** осуществляется длительным (более 2 сек) нажатием кнопки 

. Переход из режима **run** в режим **stop** осуществляется длительным (более 2 сек) нажатием кнопки 

. В режиме **stop** допускается изменение параметра «Уставка 1». В режиме **run** параметр заблокирован от изменения по сети и с передней

панели контроллера. Параметр «Гистерезис 1» остается доступным для редактирования в обоих режимах.

Пример на языке ST выглядит следующим образом:

```
PROGRAM PLC_PRG
VAR
    tm1, tm2,tm:TON;
    starting:BOOL:=FALSE;
    param_ptr:POINTER TO parameter_descriptor;
    running:BOOL:=FALSE;
END_VAR
VAR CONSTANT
    key_start:BYTE:=32;     (*ПУСК/СТОП*)
    key_stop:BYTE:=64;      (*ВЫХОД*)
END_VAR
IF NOT starting THEN            (*выполняется при начальной загрузке контроллера*)
     SetWorkScreenCount(1);  (*установим количество экранов равное 1*)
     ClearScreen(0);              (*очистка экрана*)
     starting:=TRUE;     (*установим атрибут, чтобы больше не входить в этот цикл*)
END_IF
tm(in:=kbrd=key_start AND NOT running,pt:=t#2s);       (*если пользователь нажал
                        кнопку ПУСК/СТОП более чем на 2 сек и прибор не запущен*)
IF tm.Q THEN                (*условие для выполнения этого цикла*)
      running:=TRUE;       (*установим флаг что теперь прибор находится в режиме
                                                                    работа*)
      param_ptr:=READ_BY_HASH(16#37B7,0);    (*чтение дескриптора параметра*)
      param_ptr^.flags.0:=FALSE; (*запрет записи параметра "Уставка 1" с передней
                                                                    панели*)
      param_ptr^.flags.1:=FALSE;     (*запрет записи параметра "Уставка 1" по сети*)
      SAVE_PAR_ATTRIB(param_ptr);                (*запись атрибутов параметра*)
END_IF

tm1(in:=kbrd=key_stop AND running,pt:=t#2s); (*если пользователь нажал кнопку
                                ПУСК/СТОП более чем на 2 сек и прибор запущен*)
IF tm1.Q THEN                                (*условие для выполнения этого цикла*)
      running:=FALSE;
      param_ptr:=READ_BY_HASH(16#37B7,0);   (*чтение дескриптора параметра*)
      param_ptr^.flags.0:=TRUE;        (*разрешение записи параметра "Уставка 1" с
                                                                    передней панели*)
      param_ptr^.flags.1:=TRUE;  (*разрешение записи параметра "Уставка 1" по
сети*)
      SAVE_PAR_ATTRIB(param_ptr);        (*запись атрибутов параметра*)
END_IF
tm2(in:=NOT tm2.Q,  pt:=t#200ms);      (*установка цикла индикации –- 200 мс*)
IF tm2.Q THEN                          (*цикл будет запущен каждые 200 мс*)
       IF running THEN                 (*если сейчас прибор запущен*)
           ClearScreen(0);             (*очищаем дисплей*)
           ShowString(0,0,0,'Run');    (*выводим информацию: прибор запущен*)
       ELSE                           (*если прибор остановлен*)
           ClearScreen(0);             (*очищаем дисплей*)
           ShowString(0,0,0,'Stop');   (*выводим информацию: прибор остановлен*)
       END_IF;
END_IF
```

Сохранение значения параметра (SAVE_PARAMETER)

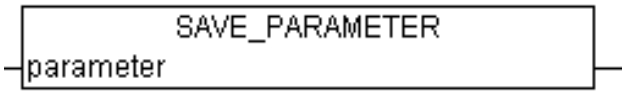


Рисунок 2.5 – Структурная схема

Таблица 2.5

Имя программного компонента	SAVE_PARAMETER		
Тип программного компонента	Функциональный блок ☐	Функция ☒	Программа ☐
Особенности работы	Для работы не требуется установка в проекте дополнительных библиотек		
Применение на контроллерах	ПЛК63, ПЛК73, ПЛК410		

Выходная переменная:	**Тип данных**	**Пояснения**
Save_parameter	INT	Функция ничего не возвращает

Описание работы

Функция записывает значение конфигурационного параметра: parameter^.value.

Внимание!

1 Все остальные значения в записываемой структуре parameter_descriptor должны полностью соответствовать редактируемому параметру.

2 Запрещается использование функции для редактирования оперативного параметра.

Применение данной функции показано на примере ее вызова в составе программы, которая при отпускании нажатых кнопок 

+  записывает значение параметра «Уставка 1», увеличенное на 1.

Пример на языке ST выглядит следующим образом:

```
PROGRAM PLC_PRG
VAR
	tm:TON;
	fm:F_TRIG;
	starting:BOOL:=FALSE;
	param_ptr:POINTER TO parameter_descriptor;
	param_adr: POINTER TO REAL;
	param_adr2: POINTER TO ARRAY[0..5] OF BYTE;
	temp: REAL;
	temp2: ARRAY [0..15] OF BYTE;
END_VAR
VAR CONSTANT
	key:BYTE:=136;
END_VAR
```

```
IF NOT starting THEN            (*выполняется при начальной загрузке контроллера*)
SetWorkScreenCount(1);          (*установим количество экранов равное 1*)
ClearScreen(0);                 (*очистка экрана*)
starting:=TRUE;             (*установим атрибут, чтобы больше не входить в этот цикл*)
END_IF

fm(clk:=kbrd=key);          (*если пользователь нажал кнопку АЛЬТ + стрелка вверх*)
IF fm.Q THEN                    (*и отпустил, то будет выполнен этот цикл*)
    param_ptr:=READ_BY_HASH(16#37B7,0);    (*чтение дескриптора параметра*)
    temp:=sp1+1.0;              (*в переменную temp занесем значение уставки + 1*)
    param_adr:=ADR(temp);                  (*в param_adr – адрес temp*)
    param_adr2:=param_adr;          (*сделаем так, чтобы переменные param_adr и
                            param_adr2 ссылались на одно и то же место в памяти*)
    temp2:=param_adr2^;         (*в temp2 занесем результат, считанный по адресу
                                 param_adr2 – фактически в temp2 попадает
                                значение из temp с учетом преобразования типов*)
    param_ptr^.value:=temp2;  (*значение переменной в param_ptr занесем temp2*)
    SAVE_PARAMETER(param_ptr);    (*запись параметра в память контроллера*)
END_IF

tm(in:=NOT tm.Q,  pt:=t#200ms);          (*установка цикла индикации – 200 мс*)
IF tm.Q THEN                             (*цикл будет запущен каждые 200 мс*)
    ShowReal(0,0,0,'%2.1f',sp1);         (*отображаем значение уставки*)
END_IF
```

Лист изменений в версиях документа

Номер версии	Дата выпуска	Содержание изменений
01	16.06.2010	Новый документ