_____ ЗАО "ЭМИКОН"_____

Интегрированная система разработки прикладного программного обеспечения CONT-Designer for Windows

Версия 2.xx

Руководство программиста

2007 г.

Содержание.

1. Введение.

Интегрированная система CONT-Designer разработана и реализована фирмой "ЭМИКОН" и предназначена для разработки прикладного программного обеспечения контроллеров серий ЭК-2000 и DCS-2000.

Настоящий документ содержит описание интегрированной системы разработки (ИСР) прикладного программного обеспечения (ПО) CONT-Designer, которая включает пользовательский язык программирования (ЯП) CONT, библиотеку прикладных функций, набор сервисных программ, а также интегрированную среду, содержащую текстовый редактор, транслятор и систему отладки.

Описания входящих в ИСР ПО CONT-Designer библиотек функций и драйверов, наиболее часто используемых при написании технологических программ, содержатся в файлах с расширением .DOC в каталоге …\WinCont\Doc.

2. Пользовательский язык программирования CONT.

Язык программирования CONT является проблемно-ориентированным языком структурированного текста, содержащим специальные операторы и структуры данных, которые отражают специфику задач управления технологическими процессами (для исключения зависания программы и подачи на дискретные выходы ложных сигналов, обеспечения корректного входа в программу после ее сброса и т.д.).

Синтаксис ЯП CONT позволяет при создании ПО использовать методы структурного, процедурного и модульного программирования, применение которых позволяет легко понимать структуру программы, осуществлять ее отладку и модификацию. Для обеспечения еще большей ясности структуры алгоритма, лежащего в основе программы, в ЯП CONT используются внутренние и внешние комментарии.

Возможность использования различных типов операций (логических, арифметических, операций отношения, операций над битами, операций извлечения адреса и номера переменных) позволяет составлять сложные выражения.

Важной особенностью задач управления технологическими процессами является необходимость гибкого переключения задач и оперативного реагирования на возникновение различных ситуаций. Также часто возникает необходимость в параллельном выполнении нескольких задач. Для удовлетворения этим требованиям ЯП CONT имеет механизм программного и аппаратного прерывания обычного хода выполнения программ при наступлении некоторого события.

Функциональные возможности ЯП **CONT** значительно расширяются благодаря включению набора библиотек, содержащих объектные коды наиболее часто используемых в технологических программах стандартных функций и драйверов, реализующих арифметику с плавающей точкой, ПИД- и ПДД2-регуляторы, фильтрацию, линеаризацию, масштабирование измеренных значений, инициализацию и чтение таймера реального времени, работу с архивами и массивами переменных, с модулями аналогового ввода-вывода и другие возможности. Открытый формат оформления функций и драйверов позволяет пользователям ЯП **CONT** самостоятельно расширять его возможности путем написания собственных функций и драйверов для дальнейшего использования их в программе.

2.1. Типы данных.

В ЯП CONT поддерживаются переменные следующих типов:
- порты (порты модулей ввода-вывода дискретных сигналов);
- биты (разряды портов ввода-вывода дискретных сигналов);
- регистры (16-разрядные ячейки энергонезависимого ОЗУ контроллера, содержащие переменные, запись в которые возможна также с помощью панели оператора в ходе выполнения программы);
- флаги (разряды регистров);
- таймеры (переменные, предназначенные для отсчета временных интервалов или измерения длительности того или иного события).

Также в ЯП CONT включена поддержка символических констант.

2.1.1. Порты ввода и вывода дискретных сигналов.

Контроллеры серии ЭК-2000 содержат набор разнообразных устройств связи с объектом (УСО), в том числе и модулей ввода и вывода дискретных сигналов. В табл. 1 приведен перечень модулей ввода и вывода дискретных сигналов. Для задания конфигурации контроллера, в том числе определения используемых УСО, используется команда меню «Проект|Конфигурация».

Таблица 1

Наименование модуля	Обозначение модуля
Модуль вывода дискретных сигналов постоянного тока	DO-01A
Модуль вывода дискретных сигналов постоянного тока	DO-03A
Модуль вывода дискретных сигналов постоянного тока	DO-04A
Модуль ввода дискретных сигналов постоянного тока	DI-01A
Модуль ввода дискретных сигналов постоянного тока	DI-03A
Модуль ввода дискретных сигналов постоянного тока	DI-04A
Модуль ввода/вывода дискретных сигналов постоянного тока	DIO-01A
Модуль ввода/вывода дискретных сигналов постоянного тока	DIO-03A
Модуль ввода/вывода дискретных Сигналов постоянного тока	DIO-04A

Каждый модуль дискретного ввода и вывода содержит 8-ми разрядные порты. Функциональные обозначения портов и их порядковые номера для каждого типа модуля дискретного ввода и вывода приведены в табл.2.

Таблица 2

Название модуля	Обозначение порта	Функциональное назначение порта
DO-01A, DO-03A	0 1 2 3 4 5	порт вывода порт вывода порт вывода порт вывода порт вывода порт чтения статуса
DI-01A, DI-03A	0 1 2 3	порт ввода порт ввода порт ввода порт ввода
DIO-01A, DIO-03A	0 1 2 3 4 5	порт ввода порт ввода порт вывода порт вывода порт вывода порт ввода
DI-04A	0 1 2 3 4 5	порт ввода порт ввода порт ввода порт ввода порт ввода порт ввода
DO-04A	0 1 2 3 4 5	порт вывода порт вывода порт вывода порт вывода порт вывода порт вывода
DIO-04A	0 1 2 3 4 5	порт ввода порт ввода порт ввода порт вывода порт вывода порт вывода

Для доступа к портам ввода и вывода дискретных сигналов в пользовательской программе используются их имена. Для определения портов ввода и вывода дискретных сигналов используется команда меню «Проект|Конфигурация».

Для получения порядкового номера порта в модуле в пользовательской программе перед именем порта необходимо указать специальную унарную операцию взятия номера "#", например: #ПОРТ_1.

Доступ к портам других типов модулей напрямую недоступен, обращение к ним осуществляется с помощью библиотечных функций.

2.1.2. Дискретные входы/выходы.

Дискретные входы/выходы (биты) представляют собой разряды портов ввода и вывода дискретных сигналов. Дискретные выходы могут включаться и выключаться операторами языка CONT или отладочными средствами для управления исполнительными механизмами.

Для доступа к битам в пользовательской программе используются их имена. Для определения битов используется команда меню «Проект | Конфигурация». Для получения порядкового номера бита в порте в пользовательской программе перед именем бита необходимо указать специальную унарную операцию взятия номера "#", например: #ВХОД_1.

2.1.3. Регистры.

Регистры представляют собой 16-разрядные ячейки памяти контроллера, в которых хранятся технологические параметры и промежуточные данные, используемые пользовательской программой.

Для доступа к регистрам в пользовательской программе используются их имена. При определении регистров в окне данных (на странице «Регистры и флаги»), им присваиваются как имена, так и порядковые номера, что позволяет закреплять за ними фиксированные адреса памяти и организовывать массивы регистров. Для получения порядкового номера регистра в пользовательской программе перед его именем необходимо указать специальную унарную операцию взятия номера "#" (например, #ИЗМЕРЕНИЕ_1), а для получения абсолютного адреса (в байтах относительно сегмента данных - унарную операцию взятия адреса "$" (например, $ИЗМЕРЕНИЕ_1).

Помимо регистров, назначенных пользователем, в программе могут быть использованы системные регистры STATUS и RUNTIMEERROR.

Регистр STATUS располагается первым в области регистров и модифицируется операционной системой. Разрядами этого регистра являются системные флаги. Данный регистр со своими флагами автоматически добавляется к списку переменных при создании нового проекта.

Далее в области регистров располагается регистр RUNTIMEERROR, который так же модифицируется операционной системой и автоматически добавляется к списку переменных при создании нового проекта. Этот регистр содержит код ошибки выполнения пользовательской программы.

2.1.4. Флаги.

Флаги представляют собой разряды внутри регистров, которые могут взводиться и сбрасываться операторами языка CONT, отладочными средствами и операционной системой. Флаги регистрируют какие-либо события, возникающие при выполнении пользовательской программы. Значение флага может оцениваться в операторах цикла и условного перехода, при обработке прерываний или в выражениях.

В пользовательских программах могут использоваться шестнадцать системных флагов, которые модифицируются операционной системой и являются разрядами системного регистра STATUS. Нулевой разряд - флаг ДЕЛЕНИЕ_НА_НУЛЬ взводится, если при делении частное превышает максимально возможное значение (например, если делитель равен нулю). Первый разряд - флаг ПЕРЕПОЛНЕНИЕ взводится, если в результате сложения, вычитания или умножения возникает переполне-

ние. При использовании арифметических операций в пользовательской программе необходимо предусмотреть средства (например, программные прерывания), проверяющие значения данных флагов, и реагирующие соответствующим образом, если хотя бы один из этих флагов взведен. Сброс взведенных системных флагов должен осуществляться в пользовательской программе.

Для доступа к флагам в пользовательской программе используются их имена. Для определения флагов необходимо открыть в окне данных страницу «Регистры и флаги» и, после выбора нужного регистра и нажатия правой кнопки мыши, выбрать команду меню «Окно флагов». Для получения номера разряда регистра в пользовательской программе перед именем флага необходимо указать специальную унарную операцию взятия номера "#", например: #СОБЫТИЕ_1.

2.1.5. Таймеры.

Таймеры представляют собой структуры данных, предназначенные для отсчета временных интервалов или измерения длительности того или иного события. Каждый таймер имеет регистр для хранения уставки, счетчик временных интервалов, входной бит управления "старт/стоп" ("1/0") и выходные биты состояния - "счет/нуль" ("1/0") и "сработал/не сработал" ("1/0").

Перед запуском таймера в его регистр и счетчик необходимо записать начальное значение (уставку), а также задать один из двух режимов работы. Указанные действия выполняются командой ЗАГРУЗИТЬ. Емкость счетчика таймера - 16 двоичных разрядов, максимальное значение записываемой уставки - 65535.

Запуск таймера на счет выполняется командой СТАРТ. После запуска таймера его биты "старт/стоп" и "счет/нуль" устанавливаются в состояние "1", бит "сработал/не сработал" остается в состоянии "0". При этом уставка перезагружается из регистра таймера в его счетчик (режим 0), или текущее значение счетчика таймера остается неизменным (режим 1). Значение таймерного счетчика начинает декрементироваться через каждые 10 милисекунд.

Как только значение счетчика достигнет нуля, его биты старт/стоп" и "счет/нуль" устанавливаются в состояние "0", а бит "сработал/не сработал" - в состояние "1". При считывании бита "сработал/не сработал" в пользовательской программе, его значение обнуляется.

Принудительный останов таймера выполняется командой СТОП. При этом значение его счетчика прекращает изменяться, бит "старт/стоп" устанавливается в состояние "0", бит "счет/нуль" остается в состоянии "1", а бит "сработал/не сработал" - в состоянии "0".

Для чтения значений таймерных счетчиков в пользовательской программе используются их имена. Для доступа к таймерным битам управления и состояния перед именем или после него используется специальный символ "@". При определении таймеров в окне данных (на странице «Таймеры»), им присваиваются как имена, так и порядковые номера, что позволяет закреплять за ними фиксированные адреса памяти и организовывать массивы таймеров. Для получения порядкового номера таймера в пользовательской программе перед его именем необходимо указать специальную унарную операцию взятия номера "#" (например, #ТАЙМЕР_1), а для получения абсолютного адреса (в байтах относительно сегмента данных - унарную операцию взятия адреса "$" (например, $ТАЙМЕР_1).

2.1.6. Символические константы.

Символические константы представляют собой 16-разрядные ячейки ППЗУ контроллера, в которых хранятся неизменяемые в программе величины.

При определении символических констант в окне данных (на странице «Константы»), им присваиваются как имена, так и значения. Значения символических констант, присвоенные на этапе написания программы, доступны только для чтения и не изменяются во время выполнения программы. Для доступа к значениям символических констант в пользовательской программе используются их имена.

Заданный в окне данных список следующих друг за другом символических констант считается массивом констант. Имеется возможность обращаться к элементу массива символических констант, указывая начало массива и номер элемента в нем. Для получения номера символической константы перед ее именем необходимо указать специальную унарную операцию взятия номера "#" (например, #УСТАВКА_1). Для копирования массив символических констант в массив регистров используется стандартная библиотечная функция CONCPY.

2.2. Зарезервированные слова.

Зарезервированные слова используются для обозначения операторов и псевдооператоров ЯП CONT. Они записывается заглавными буквами и не могут использоваться в качестве идентификаторов. В табл. 3 перечислены все зарезервированные слова ЯП CONT.

Таблица 3

БИБЛИОТЕКА	ЗАГРУЗИТЬ	ПОКА
ВЗВЕСТИ	ЗАПРЕТИТЬ	ПРЕРЫВАНИЕ
ВКЛЮЧИТЬ	ЗНАЧЕНИЕ	ПРОЦЕДУРА
ВЫЗВАТЬ	ИДТИ	РАЗРЕШИТЬ
ВЫКЛЮЧИТЬ	ИНАЧЕ	РЕСТАРТ
ВЫПОЛНЯТЬ	ОЖИДАТЬ	СБРОСИТЬ
ВЫХОД	ОСТАЛЬНЫЕ	СТАРТ
ВЫЧИСЛИТЬ	ПЕРЕЙТИ	СТОП
ДЕБЛОКИРОВАТЬ	ПЕРЕСЛАТЬ	ФУНКЦИЯ
ДРАЙВЕР	ПОВТОРЯТЬ	ЦИКЛ
ЕСЛИ	ПОДПРОГРАММА	

2.3. Идентификаторы.

Идентификаторы в ЯП CONT - это имена переменных и символических констант, заданных в окне данных, а также меток, процедур, подпрограмм обслуживания прерываний, функций и драйверов. Максимальная длина идентификатора - 25 символа. Символами, образующими идентификатор, могут быть заглавные русские и латинские буквы, цифры, а также знак подчеркивания "_".

Примеры правильных идентификаторов:

N, УСТАВКА_10001.

Примеры неправильных идентификаторов:
ВЫХОД (является ключевым словом),
ИЗМЕРЕНИЕ#100 (содержит недопустимый символ),
КОНЦЕВИК 1000 (содержит пробел).

2.4. Числовые константы.

В качестве числовых констант в ЯП CONT могут использоваться десятичные, двоичные и шестнадцатеричные целые числа. Этим они отличаются от символических констант, которые являются идентификаторами ЯП CONT, т.е. обозначаются в программе своими именами. Еще одно различие: под числовые константы память не выделяется, а символические константы размещаются в ППЗУ контроллера. Таким образом, числовые константы предпочтительнее тем, что под них не выделяется память контроллера. Символические константы предпочтительнее тем, что присвоение им новых значений в окне данных не приводит к изменению текста программы. Кроме того, имеется возможность обращаться к элементу массива символических констант, указывая начало массива и номер элемента в нем.

Если числовая константа является двоичным числом, то она должна заканчиваться латинской буквой "B", если шестнадцатеричным числом - то латинской буквой "H".

2.5. Операции и выражения.

Выражение задает порядок выполнения действий над элементами данных и состоит из операндов, операций, круглых скобок и внутренних комментариев.

Операнды, входящие в выражение, могут быть именами регистров, таймеров, портов, флагов, дискретных входов/выходов, таймерных битов, констант, элементов массивов регистров и таймеров, а также числовых и символических констант. Имена переменных и символических констант не должны начинаться с цифр. Таймерный бит "старт/стоп" обозначается именем таймера, перед которым стоит символ '@' (пример: @ТАЙМЕР). Таймерный бит "счет/нуль" обозначается именем таймера, за которым стоит символ '@' (пример: ТАЙМЕР@). Таймерный бит "сработал/не сработал" обозначается именем таймера, заключенным между символами '@' (пример: @ТАЙМЕР@).

Операции определяют действия, которые надо выполнить над операндами. Круглые скобки ставятся для управления порядком выполнения операций.

В выражениях ЯП CONT используются следующие операции:

- унарные: # (взятие номера), $ (взятие адреса), - (арифметическое отрицание), ! (логическое отрицание), ~(инверсия);
- мультипликативные: * (умножение), / (деление), : (беззнаковое деление), % (вычисление остатка от беззнакового деления), << (сдвиг влево), >> (сдвиг вправо), & (логическое и арифметическое И);
- аддитивные: + (сложение), - (вычитание), | (логическое и арифметическое ИЛИ), ^ (логическое и арифметическое ИСКЛЮЧАЮЩЕЕ ИЛИ);
- отношения: == (равно), != (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).

Приоритет операции определяется ее типом: унарные операции имеют наивысший приоритет, далее следуют мультипликативные операции, затем – аддитивные, и, наконец – операции отношения.

Для определения старшинства операций используются следующие правила:

1. Операнд, находящийся между двумя операциями с различными приоритетами, связывается с операцией, имеющей более высокий приоритет.
2. Операнд, находящийся между двумя операциями с равными приоритетами, связывается с той операцией, которая находится слева.
3. Выражение, заключенное в скобки, перед выполнением вычисляется, как отдельный операнд.
4. Операции с равным приоритетом производятся слева направо с возможным регулированием порядка выполнения скобками.

Для упрощения чтения программы между операндами и операциями можно вводить внутренние комментарии, которые могут содержать строчные буквы, пробелы и символы табуляции, а также следующие символы: , . ? '.

Унарная операция «!» (логическое НЕ) и другие логические операции применяются только для логических подвыражений и битовых переменных (дискретных входов/выходов, флагов и таймерных битов). Результатом выполнения любой логической операции является логическое значение «1» (истина) или «0» (ложь).

Список логических операций и результаты их действия приведены в табл.4.

Таблица 4

Операция	Действие	Выражение	A	B	Результат
!	Логическое НЕ	!A	1 0		0 1
&	Логическое И	A & B	1 1 0 0	1 0 1 0	1 0 0 0
|	Логическое ИЛИ	A | B	1 1 0 0	1 0 1 0	1 1 1 0
^	Логическое исключающее ИЛИ	A ^ B	1 1 0 0	1 0 1 0	0 1 1 0

Арифметические операции и унарная операция «~» (инверсия) применяются для любых подвыражений, переменных и констант.

Результатом выполнения любой арифметической операции является 16-разрядное слово данных. Результат не должен выходить за пределы диапазона -32768…32767, в противном случае взводится системный флаг ПЕРЕПОЛНЕНИЕ.

Список арифметических операций и операций над битами, а также результаты их действия приведены в табл.5.

Таблица 5

Операция	Действие	Выраже-ние	A	B	Результат
+	Сложение	A + B	500	200	700
-	Вычитание	A - B	700	200	500
*	Умножение	A * B	50	70	3500
/	Деление	A / B	-3510	70	-50
:	Беззнаковое деление	A : B	FFFFH	2	32767
%	Остаток от беззнакового деления	A % B	FFFFH	2	1
<<	Сдвиг влево	A << B	2	3	16
>>	Сдвиг вправо	A >> B	64	3	8
~	Инверсия	~A	1101B		0010B
&	Арифметическое И	A & B	1101B	0101B	0101B
|	Арифметическое ИЛИ	A | B	1101B	0101B	1101B
^	Арифметическое ИСКЛЮЧАЮЩЕЕ ИЛИ	A ^ B	1101B	0101B	1000B

Операции отношения применяются для любых подвыражений, переменных и констант. Результатом выполнения любой операции отношения является логическое значение «1» (истина) или «0» (ложь).

Список операций отношения и результаты их действия приведены в табл.6.

Таблица 6

Операция	Действие	Выражение	A	B	Результат
<	Меньше	A < B	10 10 10	5 10 50	0 0 1
<=	Меньше или равно	A <= B	10 10 10	5 10 50	0 1 1
>	Больше	A > B	10 10 10	5 10 50	1 0 0
>=	Больше или равно	A >= B	10 10 10	50 10 50	1 1 0
==	Равно	A == B	10 10 10	5 10 50	0 1 0
!=	Не равно	A != B	10 10 10	5 10 50	1 0 1

Операция взятия номера «#» применяется для регистров, флагов, таймеров, портов дискретного ввода-вывода, дискретных входов/выхо-

дов и символических констант.

Операция взятия адреса «$» применяется для регистров и таймеров и позволяет получить адрес переменной относительно сегмента данных контроллера.

Ниже приведены примеры использования выражений в ЯП CONT.

Пример 1:

```
Ф_ВЫХОД_ЗА_УСТАВКУ =
(ЗНАЧ_ПАРАМЕТРА[НОМЕР_ПАРАМЕТРА] < УСТАВКА_МИН[НОМЕР_ПАРАМЕТРА]) |
(ЗНАЧ_ПАРАМЕТРА[НОМЕР_ПАРАМЕТРА] > УСТАВКА_МАКС[НОМЕР_ПАРАМЕТРА]);
```

Пример 2:

```
ЧИСЛО_ВКЛЮЧ_НАСОСОВ = (СОСТОЯНИЕ_НАСОСА[0] == ВКЛЮЧЕН)
                     + (СОСТОЯНИЕ_НАСОСА[1] == ВКЛЮЧЕН)
                     + (СОСТОЯНИЕ_НАСОСА[2] == ВКЛЮЧЕН);
```

Пример 3:

```
РЕГИСТР1 = РЕГИСТР2 > ЗНАЧЕНИЕ + РЕГИСТР3;
```

В примере 3 сначала рассчитывается сумма (ЗНАЧЕНИЕ + РЕГИСТР3), которая затем сравнивается с переменной РЕГИСТР2, поскольку приоритет операции сложения выше, чем операции сравнения. Если необходимо сравнить РЕГИСТР2 и ЗНАЧЕНИЕ, а затем сложить результат, равный «0» или «1» с переменной РЕГИСТР3, необходимо изменить выражение с помощью круглых скобок следующим образом:

```
РЕГИСТР1 = (РЕГИСТР2 > ЗНАЧЕНИЕ) + РЕГИСТР3;
```

Пример 4:

```
РЕГИСТР1 = ДИСКРЕТНЫЙ_ПОРТ_СТ << 8 | ДИСКРЕТНЫЙ_ПОРТ_МЛ;
```

В примере 4 выполняется запись двух портов дискретного ввода в старший и младший байты регистра.

Пример 5:

```
@ЧИСЛО_ИНТЕРВАЛОВ = ! бит состояния ТАЙМЕР1@ &
                    ! бит состояния ТАЙМЕР2@ &
                    ! флаг Ф_СОСТОЯНИЕ;
```

Здесь таймер ЧИСЛО_ИНТЕРВАЛОВ запускается (как по команде СТАРТ), если одновременно выполняются три условия:

- таймер ТАЙМЕР1 досчитал до 0, т.е. его бит состояния имеет значение "нуль";
- таймер ТАЙМЕР2 также досчитал до 0;
- сброшен флаг Ф_СОСТОЯНИЕ.

Если одно из этих условий не выполняется, таймер ЧИСЛО_ИНТЕРВАЛОВ останавливается (как по команде СТОП).

```
Пример 6:

Ф_СОСТОЯНИЕ1 = содержимое флага Ф_СОСТОЯНИЕ2;
```

Поскольку логическое выражение состоит только из одного операнда, то действие оператора присваивания состоит в простом копировании содержимого флага Ф_СОСТОЯНИЕ2 в флаг Ф_СОСТОЯНИЕ1.

2.6. Разделители.

В ЯП CONT используются следующие разделители:

() ; [] " /* */

Круглые скобки () используются для выделения операндов внутри операторов и псевдооператоров, а также для изменения последовательности вычисления выражений.

Точка с запятой ";" завершает операторы, псевдооператоры и выражения.

Квадратные скобки [] используются для выделения метки.

Двойные кавычки ", а также пара ограничителей /* и */ служат для выделения внешних комментариев.

Пример использования разделителей:

```
/* Использование метки КОНЕЦ_ПРОВЕРКИ в качестве
 операнда оператора безусловного перехода */

ИДТИ на метку (КОНЕЦ_ПРОВЕРКИ);

" Здесь располагаются какие-либо операторы программы:
".....................................................
" Определяется метка КОНЕЦ_ПРОВЕРКИ

[КОНЕЦ_ПРОВЕРКИ]
```

2.7. Комментарии.

Внутри операторов между зарезервированными словами и операндами, а также внутри выражений между операндами и операциями можно записывать в произвольном порядке внутренние комментарии - не являющиеся зарезервированными слова, содержащие строчные буквы, пробелы и символы табуляции, а также следующие символы: , . ? '.

Для более подробного описания структуры программы используются внешние комментарии - фрагменты текста, находящиеся между операторами, которые выделяются либо двойными кавычками ("), либо двумя специальными ограничителями (/* и */).

Двойные кавычки удобно использовать, если комментарий располагается лишь на одной строке. Они указывают компилятору, что вся информация на данной строке, находящаяся справа от двойных кавычек, должна рассматриваться как комментарий. После двойных кавычек могут стоять любые символы.

Ограничители /* и */ удобно использовать, если комментарий занимает несколько строк, или фрагмент строки программы. Они указывают компилятору, что вся информация, находящаяся внутри ограничителей, должна рассматриваться как комментарий. Внутри ограничителей могут стоять любые символы, кроме сочетаний /* и */. Таким обра-

зом, вложенные комментарии, использующие ограничители /* и */, в ЯП CONT запрещены.

Внутренние и внешние комментарии игнорируются компилятором и поэтому никакого влияния на выполнение программы не оказывают.

Внутренние и внешние комментарии позволяют программисту писать технологические программы, которые легко читать, отлаживать и модифицировать. При написании программ может использоваться терминология того оборудования, для которого предназначен контроллер. Использование внешних комментариев удобно также при отладке программ, когда требуется временно исключить выполнение какой-либо части программы.

Следующий фрагмент технологической программы, написанной на ЯП CONT, иллюстрирует использование внутренних и внешних комментариев:

```
/*
Символическом константам ВЕРХНИЙ_ПРЕДЕЛ и НИЖНИЙ_ПРЕДЕЛ
соответствуют +50 градусов и -50 градусов по Цельсию.
Флаги ВЫХОД_ЗА_НИЖНИЙ_ПРЕДЕЛ и ВЫХОД_ЗА_ВЕРХНИЙ_ПРЕДЕЛ
обнулены.
*/
ЕСЛИ показания датчика ТЕРМОМЕТР <= ВЕРХНИЙ_ПРЕДЕЛ, то
{/* Температура не превышает верхний предел
    Проверить, выходит ли температура за нижний предел */
 ЕСЛИ показания датчика ТЕРМОМЕТР < НИЖНИЙ_ПРЕДЕЛ, то
 {  /* Температура выходит за нижний предел */
    ВЫХОД_ЗА_НИЖНИЙ_ПРЕДЕЛ = 1;
    " Включить нагреватель
    ПОВЫШЕНИЕ_ТЕМПЕРАТУРЫ = 1;
 }
 /* Температура находится внутри допустимого диапазона,
    автоматический переход на метку КОНЕЦ_ПРОВЕРКИ */
}
ИНАЧЕ температура превышает верхний предел
{
 ВЫХОД_ЗА_ВЕРХНИЙ_ПРЕДЕЛ = 1;
 " Включить вентилятор
 ПОНИЖЕНИЕ_ТЕМПЕРАТУРЫ = 1;
}
[КОНЕЦ_ПРОВЕРКИ]
```

2.8. Операторы.

В ЯП CONT включены следующие операторы, характерные для большинства универсальных языков высокого уровня:

- оператор присваивания;
- оператор условного перехода;
- оператор безусловного перехода;
- оператор-переключатель;
- оператор вычислений;
- оператор цикла с предусловием;
- оператор цикла с постусловием;
- оператор цикла со счетчиком;
- операторы работы с процедурами;
- оператор принудительного завершения;
- оператор вызова функций;
- оператор ожидания.

Для управления переменными специальных типов в ЯП CONT имеются следующие дополнительные операторы:

- операторы записи в дискретные выходы;
- операторы записи во флаги;
- операторы управления таймерами.

Помимо перечисленных в ЯП CONT используется также ряд специальных операторов, введение которых связано со спецификой создания прикладного ПО систем автоматизации:

- оператор перезапуска;
- оператор снятия блокировки;
- операторы работы с программными и аппаратными прерываниями.

Большинство операторов ЯП CONT содержат операнды, которые выделяются круглыми скобками.

Некоторые операторы включают в себя составные операторы. Составной оператор - это последовательность произвольных операторов программы, заключенная в операторные скобки. В ЯП CONT операторными являются фигурные скобки. Составные операторы - важный инструмент ЯП CONT, позволяющий использовать при создании технологических программ метода структурного программирования.

Все операторы, не включающие в себя составные операторы, должны заканчиваться разделителем ";".

Каждый оператор может располагаться как на одной строке, так и на нескольких строках, и наоборот, на одной строке может быть как один оператор, так и несколько операторов.

Значения переменных во всех операторах, кроме ОЖИДАТЬ и ЗАГРУЗИТЬ, принимаются как целочисленные СО ЗНАКОМ (-32768... 32767). Значение переменной в качестве операнда в операторе ОЖИДАТЬ или уставки в операторе ЗАГРУЗИТЬ принимается как целочисленное БЕЗ ЗНАКА (0...65535). Значения портов во всех операторах принимаются как однобайтовые БЕЗ ЗНАКА (0...255).

2.8.1. Оператор присваивания.

Формат оператора присваивания:

Операнд_приемник = Выражение;

Операнд_приемник может быть именем регистра, элемента массива регистров, порта дискретного вывода, дискретного выхода, флага или бита управления таймера "старт/стоп". За этим операндом может следовать внутренний комментарий.

Для обеспечения совместимости с ранее написанными программами в ЯП CONT поддерживается также оператор пересылки данных:

ПЕРЕСЛАТЬ (операнд_источник) (операнд_приемник);

операнд_источник:
- регистр,
- флаг,
- таймерный счетчик,
- таймерный бит управления,
- таймерные биты состояния,
- порт,
- дискретный вход/выход,
- числовая или символическая константа в диапазоне -32768…32767 (0…FFFFH), если операнд_приемник является регистром, и в диапазоне 0…255 (0…FFh), если операнд_приемник является

портом.

Операнд_приемник в случае старой формы записи:

- регистр,
- флаг,
- порт, доступный для записи,
- дискретный выход.

Примеры использования оператора:

```
/* Новый формат: */
ВЫХОД_ЗА_УСТАВКУ[НОМЕР_ПАРАМЕТРА] =
(ЗНАЧ_ПАРАМЕТРА[НОМЕР_ПАРАМЕТРА] < УСТАВКА_МИН[НОМЕР_ПАРАМЕТРА]) |
(ЗНАЧ_ПАРАМЕТРА[НОМЕР_ПАРАМЕТРА] > УСТАВКА_МАКС[НОМЕР_ПАРАМЕТРА]);

/* Старый формат: */
ПЕРЕСЛАТЬ значение (0) в регистр (ЗНАЧЕНИЯ_ВЫХОДОВ);
```

Если операнд_приемник является флагом, таймерным битом управления или дискретным выходом, операнд_источник также должен быть битовой переменной (дискретным входом/выходом, флагом, таймерным битом) или логическим выражением.

При пересылке данных из порта в регистр старший байт регистра обнуляется. При пересылке данных в порт передается только младший байт регистра или таймерного счетчика.

2.8.2. Операторы записи в дискретные выходы.

В ЯП CONT существуют два оператора, позволяющих включать и выключать дискретные выходы.

Формат операторов:

```
ВКЛЮЧИТЬ  (имя_дискретного_выхода);
ВЫКЛЮЧИТЬ (имя_дискретного_выхода);
```

Кроме того, для управления дискретными выходами может быть использован оператор присваивания. Если значение выражения равно нулю, дискретный выход выключается, в противном случае дискретный выход включается.

Пример использования оператора для формирования импульса:

```
ОЖИДАТЬ (5) интервалов;
КЛАПАН_1 = 1;
ОЖИДАТЬ (5) интервалов;
КЛАПАН_1 = 0;
```

2.8.3. Операторы записи во флаги.

В ЯП CONT существуют два оператора, позволяющих взводить и сбрасывать флаги.

Формат операторов:

```
ВЗВЕСТИ (имя_флага);
СБРОСИТЬ (имя_флага);
```

Кроме того, для управления флагами может быть использован оператор присваивания. Если значение выражения равно нулю, флаг сбрасывается, в противном случае флаг взводится.

Пример использования оператора:

```
ВЫХОД_ЗА_УСТАВКУ = 1;
ПРИХОД_В_НОРМУ = 0;
```

2.8.4. Операторы управления таймерами.

Перед запуском таймера в него необходимо загрузить начальное значение (уставку) и задать один из 2-х режимов его работы. Эти действия выполняет следующий оператор:

```
ЗАГРУЗИТЬ (имя_таймера) (уставка) (режим);
```

уставка: - регистр,
- порт,
- числовая или символическая константа в диапазоне 1…65535.

Если в качестве уставки используется регистр, то его значение воспринимается как целочисленное БЕЗ ЗНАКА.

режим: - числовая или символическая константа, значение которой равно 0 или 1.

Для запуска таймера используется оператор следующего формата:

```
СТАРТ (имя_таймера);
```

После запуска таймера его биты "старт/стоп" и "счет/нуль" устанавливаются в состояние "1", бит "сработал/не сработал" остается в состоянии "0". При этом уставка перезагружается из регистра таймера в его счетчик (режим 0), или текущее значение счетчика таймера остается неизменным (режим 1). Значение таймерного счетчика начинает декрементироваться через каждые 10 миллисекунд. Как только значение счетчика достигнет нуля, его биты "старт/стоп" и "счет/нуль" устанавливаются в состояние "0", а бит "сработал/не сработал" - в состояние "1".

Для останова таймера используется оператор следующего формата:

```
СТОП (имя_таймера);
```

После останова таймера значение его счетчика прекращает изменяться, бит "старт/стоп" устанавливается в состояние "0", бит "счет/нуль" остается в состоянии "1", а бит "сработал/не сработал" - в состоянии "0".

Для управления таймерами может быть также использован оператор присваивания. Если значение выражения равно нулю, таймер запускается, в противном случае таймер останавливается.

2.8.5. Оператор условного перехода.

Формат оператора:

```
ЕСЛИ выражение
   первый_составной_оператор
ИНАЧЕ
   второй_составной_оператор
```

Составляющая "ИНАЧЕ второй_составной_оператор" является не обязательной.

В случае, если условие, заданное выражением истинно, то есть результат вычисления выражения не равен нулю, выполняется первый составной оператор. Второй составной оператор в этом случае не выполняется.

Если результат вычисления выражения равен нулю, выполняется второй и не выполняется первый составной оператор.

Пример использования оператора условного перехода:

```
ЕСЛИ (ТЕМПЕРАТУРА < ВЕРХНИЙ_ПРЕДЕЛ) &
   (ТЕМПЕРАТУРА > НИЖНИЙ_ПРЕДЕЛ), то {
   Ф_ТЕМПЕРАТУРА_В_НОРМЕ = 1;
   Ф_ВЫХОД_ЗА_УСТАВКУ = 0;
}
ИНАЧЕ {
   Ф_ТЕМПЕРАТУРА_В_НОРМЕ = 0;
   Ф_ВЫХОД_ЗА_УСТАВКУ = 1;
}
```

2.8.6. Оператор безусловного перехода.

Формат оператора:

```
ИДТИ (имя_метки);
```

Пример использования оператора:

```
ИДТИ на метку (НАЧАЛО_РАБОЧЕГО_ЦИКЛА);
```

Переход на метку считается запрещенным, если она находится внутри цикла, а команда перехода - вне цикла, либо наоборот.

Оператор ИДТИ разрешается использовать для выхода из процедуры или подпрограммы обслуживания прерывания на указанную метку. В этом случае метка должна размещаться в инициализирующей части или в рабочем цикле и не должна находиться внутри цикла.

2.8.7. Оператор-переключатель.

Формат оператора:

```
ПЕРЕЙТИ (операнд) составной_оператор
```

Составной оператор здесь представляет собой набор произвольного числа блоков следующего вида:

```
ЗНАЧЕНИЕ (значение):
  последовательность_операторов
```

Составной оператор может заканчиваться блоком следующего вида:

```
ОСТАЛЬНЫЕ:
  последовательность_операторов
```

```
операнд: - регистр,
         - флаг,
         - таймерный счетчик,
         - таймерный бит управления,
         - таймерные биты состояния,
         - порт,
         - дискретный вход/выход.
```

значение - числовая или символическая константа в диапазоне -32768...32767 (0…FFFFH), если операнд является регистром или таймерным счетчиком, в диапазоне 0...255 (0…FFH), если операнд является портом, и в диапазоне 0...1, если операнд является дискретным входом/выходом, таймерным битом или флагом.

Данный оператор является обобщением оператора условного перехода и позволяет сделать выбор из произвольного числа имеющихся вариантов. Здесь под вариантом понимается возможное значение операнда, указанное после зарезервированного слова ЗНАЧЕНИЕ.

Если текущее значение операнда совпадает с одним из возможных вариантов, происходит переход на первый оператор, следующий после зарезервированного слова ЗНАЧЕНИЕ, в котором указано данное значение.

Если текущее значение операнда не совпадает ни с одним из возможных вариантов, происходит переход на первый оператор, следующий после зарезервированного слова ОСТАЛЬНЫЕ. Если зарезервированное слово ОСТАЛЬНЫЕ отсутствует, то переход осуществляется на следующую за оператором ПЕРЕЙТИ команду.

В любой последовательности операторов, следующей за зарезервированным словом ЗНАЧЕНИЕ или ОСТАЛЬНЫЕ, можно использовать оператор ВЫХОД для принудительного перехода на следующую за оператором ПЕРЕЙТИ команду.

Пример использования оператора:

```
" Если параметр РЕЗУЛЬТАТ меньше 0 или больше 2 - ошибка
" Если параметр РЕЗУЛЬТАТ равен 1 или 2 - сформировать
" соответствующее число импульсов:

СБРОСИТЬ флаг (ОШИБКА);
СБРОСИТЬ флаг (НУЛЬ);

ЕСЛИ параметр РЕЗУЛЬТАТ < 0, то {
  ВЗВЕСТИ флаг (ОШИБКА);
}
ИНАЧЕ {
  ПЕРЕЙТИ, если параметру (РЕЗУЛЬТАТ) соответствует {
  ЗНАЧЕНИЕ (0):
```

```
        ВЗВЕСТИ флаг (НУЛЬ);
        ВЫХОД;
      ЗНАЧЕНИЕ (2):
        ОЖИДАТЬ (1) интервал;
        ВКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
        ОЖИДАТЬ (1) интервал;
        ВЫКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
      ЗНАЧЕНИЕ (1):
        ОЖИДАТЬ (1) интервал;
        ВКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
        ОЖИДАТЬ (1) интервал;
        ВЫКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
        ВЫХОД;
      ОСТАЛЬНЫЕ значения:
        ВЗВЕСТИ флаг (ОШИБКА);
      }
    }
    ЕСЛИ флаг ОШИБКА != 0 {
      ВЫЗВАТЬ процедуру (ОБРАБОТКА_ОШИБКИ);
    }
```

2.8.8. Оператор вычислений.

Для выполнения вычислений в ЯП CONT применяются выражения и операторы присваивания.

Для обеспечения совместимости с ранее написанными программами в ЯП CONT поддерживается также специальный оператор вычислений:

```
ВЫЧИСЛИТЬ (операнд_1) (операция) (операнд_2) (результат);

операнд_1: - регистр,
           - флаг,
           - таймерный счетчик,
           - порт,
           - дискретный вход/выход,
           - числовая или символическая константа в диапазоне
-32768...32767 (0…FFFFH), если результат является регистром;
в диапазоне 0...255 (0…FFH), если результат является портом и
в диапазоне 0...1, если результат является флагом или дискрет-
ным входом/выходом.

операция:  + (сложение),
           - (вычитание),
           * (умножение),
           / (деление с учетом знака),
           : (беззнаковое деление),
           % (вычисление остатка от беззнакового деления),
           & (И),
           | (ИЛИ),
           ^ (ИСКЛЮЧАЮЩЕЕ ИЛИ),
           ! (инверсия),
           < (сдвиг влево),
           > (сдвиг вправо).

операнд_2: - регистр,
           - флаг,
           - таймерный счетчик,
```

- порт,
- дискретный вход/выход,
- числовая или символическая константа в диапазоне -32768...32767 (0…FFFFH), если результат является регистром; в диапазоне 0...255 (0…FFH), если результат является портом и в диапазоне 0...1, если результат является флагом или дискретным входом/выходом.

результат: - регистр,
- флаг,
- порт, доступный для записи,
- дискретный выход.

Если операция - инверсия (!) 1-ого операнда, то формат команды:

ВЫЧИСЛИТЬ (операнд_1) (!) (результат);

Если результат - флаг, или дискретный выход, то операнды должны являться флагами или дискретными входами/выходами. Допустимыми операциями в этом случае являются: &, |, ^, !.

Если при сложении, вычитании или умножении происходит переполнение разрядной сетки, то взводится системный флаг ПЕРЕПОЛНЕНИЕ. Если при делении делитель (операнд_2) равен нулю, то взводится системный флаг ДЕЛЕНИЕ_НА_НУЛЬ.

Пример использования оператора:

```
ВЫЧИСЛИТЬ сумму (ЗНАЧЕНИЕ_1) (+) (ЗНАЧЕНИЕ_2),
результат поместить в регистр (РЕЗУЛЬТАТ);
```

2.8.9. Оператор цикла со счетчиком.

Формат оператора:

```
ЦИКЛ (операнд) составной_оператор
```

операнд: - регистр,
- порт,
- числовая или символическая константа в диапазоне 1...32767.

Данный оператор обеспечивает выполнение последовательности операторов, образующих составной оператор, N раз, где N - значение операнда. При этом изменения значения самого операнда не происходит. В составной оператор может входить другой оператор цикла и т.д. Вложенность циклов не ограничена.

Запрещенными являются операторы условного и безусловного переходов, если они расположены за пределами цикла, а метка - внутри, либо наоборот, если они находятся внутри цикла, а метка - за его пределами. Для принудительного выхода из цикла используется оператор ВЫХОД.

Пример использования оператора:

```
" Сформировать последовательность из N импульсов:

ЦИКЛ со счетчиком (N) {
  ОЖИДАТЬ (1) интервал;
  ВКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
  ОЖИДАТЬ (1) интервал;
  ВЫКЛЮЧИТЬ дискретный выход (ПОДАЧА_ИМПУЛЬСА);
```

```
}
```

2.8.10. Оператор цикла с предусловием.

Формат оператора:

```
ПОВТОРЯТЬ выражение составной_оператор
```

Данный оператор обеспечивает многократное выполнение составного оператора, пока истинно условие, заданное выражением. Перед каждым выполнением составного оператора производится вычисление выражения. Если результат вычисления выражения не равен нулю, составной оператор выполняется и снова проверяется условие. Если результат вычисления выражения равен нулю, происходит выход из цикла и переход к следующему за ним оператору.

Частный случай оператора - если составной оператор пустой (т.е. содержит только фигурные скобки). В этом случае назначение оператора сводится к ожиданию невыполнения заданного условия.

В составной оператор может входить другой оператор цикла и т.д. Вложенность циклов не ограничена.

Запрещенными являются операторы условного и безусловного переходов, если они расположены за пределами цикла, а метка - внутри, либо наоборот, если они находятся внутри цикла, а метка - за его пределами. Для принудительного выхода из цикла используется оператор ВЫХОД.

Пример использования оператора:

```
"Генерация импульсов, следующих на исполнительный механизм,
"пока температура - вне допустимого диапазона:

ПОВТОРЯТЬ выполнение,
пока (ТЕМПЕРАТУРА < МИН_УСТАВКА) | (ТЕМПЕРАТУРА > МАКС_УСТАВКА)
{
  ОЖИДАТЬ (1) интервал;
  НАГРЕВАТЕЛЬ = 1;
  ОЖИДАТЬ (1) интервал;
  НАГРЕВАТЕЛЬ = 0;
}
```

2.8.11. Оператор цикла с постусловием.

Формат оператора:

```
ВЫПОЛНЯТЬ составной_оператор
ПОКА выражение;
```

Данный оператор обеспечивает многократное выполнение составного оператора, пока истинно условие, заданное выражением. Проверка условия производится после каждого выполнения составного оператора, поэтому, в отличие от оператора ПОВТОРЯТЬ, составной оператор всегда выполняется по крайней мере один раз. Если условие истинно, то есть результат вычисления выражения не равен нулю, составной оператор выполняется и снова проверяется условие. Если результат вычисления выражения равен нулю, происходит выход из цикла и переход к следующему за ним оператору.

В составной оператор может входить другой оператор цикла и т.д. Вложенность циклов не ограничена.

Запрещенными являются операторы условного и безусловного переходов, если они расположены за пределами цикла, а метка - внутри, либо наоборот, если они находятся внутри цикла, а метка - за его пределами. Для принудительного выхода из цикла используется оператор ВЫХОД.

Пример использования оператора:

```
ВЫПОЛНЯТЬ циклически {
  ДЕБЛОКИРОВАТЬ;
} ПОКА (ПОРТ_ВВОДА & 0Fh) != 0;
```

2.8.12. Операторы работы с процедурами.

Для вызова процедуры используется следующий оператор:

```
ВЫЗВАТЬ (имя_процедуры);
```

Данный оператор может находиться в любой точке инициализирующей части программы или рабочего цикла, а также входить в составной оператор другой процедуры или подпрограммы. Таким образом, может быть организован вложенный вызов процедур.

Для выделения в программе процедуры используется следующий оператор:

```
ПРОЦЕДУРА (имя_процедуры) составной_оператор
```

Данный оператор нельзя использовать внутри составных операторов. Все процедуры должны следовать после рабочего цикла программы.

Пример использования операторов:

```
[НАЧАЛО]

  ДЕБЛОКИРОВАТЬ;
  ВЫЗВАТЬ процедуру (ОБРАБОТКА_1);

ИДТИ на метку [НАЧАЛО];

" Конец рабочего цикла программы


ПРОЦЕДУРА (ОБРАБОТКА_1) {
  ВЫЗВАТЬ процедуру (ОБРАБОТКА_2);
}

ПРОЦЕДУРА (ОБРАБОТКА_2) {
  РЕЗУЛЬТАТ  =  ЗНАЧЕНИЕ_1 + ЗНАЧЕНИЕ_2,
}
```

2.8.13. Оператор принудительного завершения.

Для принудительного завершения цикла, процедуры, подпрограммы обслуживания прерывания или оператора ПЕРЕЙТИ служит следующий оператор:

```
ВЫХОД;
```

После завершения будет выполняться оператор, следующий за соответствующим составным оператором.

Пример использования оператора:

```
ЗАГРУЗИТЬ (ТАЙМЕР) уставкой (1000), режим (0);
СТАРТ (ТАЙМЕР);
ПОВТОРЯТЬ выполнение, пока флаг (СОСТОЯНИЕ == 0) {
  ЕСЛИ значение (@ТАЙМЕР@ == 1),то { ВЫХОД из цикла; }
}
```

2.8.14. Оператор вызова функций.

Функциональные возможности ИСР ПО CONT-Designer значительно расширяются благодаря вхождению в ее состав библиотек функций, наиболее часто используемых в технологических программах. Эти функции реализуют арифметику с плавающей точкой, аналоговый и импульсный ПИД-регуляторы, фильтрацию, линеаризацию, масштабирование измеренных значений, работу с архивами и массивами переменных, с модулями аналогового ввода-вывода и выполняют другие возможности.

Открытый формат оформления функций позволяет пользователям ИСР ПО CONT-Designer самостоятельно расширять ее возможности путем написания собственных функций для дальнейшего использования их в программе.

Формат оператора:

```
ФУНКЦИЯ (имя_функции),
входные  параметры ( вх_пар_1,...,вх_пар_N )
выходные параметры ( вых_пар_1,...,вых_пар_M );

вх_пар1,...,вх_пар_N: - регистр,
                      - флаг,
                      - таймерный счетчик,
                      - таймерный бит управления,
                      - таймерные биты состояния,
                      - порт,
                      - дискретный вход/выход,
                      - числовая или символическая константа
                        в диапазоне -32768...32767.

вых_пар_1,...,вых_пар_M: - регистр,
                         - флаг,
                         - порт, доступный для записи,
                         - дискретный выход.
```

Входные (или выходные), или те и другие параметры функции могут отсутствовать.

Число входных (или выходных) параметров может быть фиксированным или переменным. Если число входных (или выходных) параметров является фиксировано, то оно должно лежать в диапазоне от 0 до

254.

Если число входных (или выходных) параметров является переменным, то при компиляции отключается проверка соответствия числа входных (или выходных) параметров, указанного в заголовке, и фактического их числа в операторе ФУНКЦИЯ. Это позволяет передавать в функцию переменное число входных и получать переменное число выходных параметров. В этом случае минимальное число входных (или выходных) параметров равно одному, а максимальное общее количество входных и выходных параметров ограничивается размером стека (8 Кбайт).

Для того, чтобы использовать в программе нестандартную функцию, в инициализирующей части программы должна быть указана библиотека, в которой находится эта функция.

Пример использования оператора:

```
" Вызов стандартной функции масштабирования
" из библиотеки LIB\MATH.LIB

ФУНКЦИЯ (FZOOM)
входные параметры ( ВРЕМЕННЫЙ, 25, 32767 ),
выходные параметры ( РЕГ_ШИМ );
```

2.8.15. Оператор ожидания.

Данный оператор служит для задержки выполнения программы на указанный интервал времени. Действие оператора может быть прервано программным или аппаратным прерыванием. Выполнение подпрограммы обслуживания прерывания во время действия оператора ОЖИДАТЬ не прекращает отсчета заданного интервала времени.

Формат оператора:

```
ОЖИДАТЬ (число_интервалов);
```

Здесь число_интервалов - количество 10-милисекундных интервалов. В качестве операнда "число_интервалов" могут использоваться следующие элементы:

- регистр,
- порт,
- числовая или символическая константа в диапазоне 1...65535.

Если в качестве операнда используется регистр, то его значения воспринимаются как целое БЕЗ ЗНАКА.

Пример использования оператора:

```
ОЖИДАТЬ в течение (5) интервалов;
```

2.8.16. Оператор перезапуска.

Оператор перезапуска предназначен для передачи управления в указанную точку программы в случае кратковременного (менее 1,5 секунд) пропадания питания контроллера или срабатывания охранного

таймера при зависании программы.

Формат оператора:

```
РЕСТАРТ (имя_метки);
```

Метка, указываемая в данном операторе, должна размещаться в инициализирующей части программы или в рабочем цикле и не должна находиться внутри цикла.

При восстановлении питания программа начинает выполняться с первой исполняемой команды, если в программе отсутствует оператор РЕСТАРТ, иначе - с команды, перед которой стоит метка, заданная в операторе РЕСТАРТ. Опрератор используется в программе один раз и обычно размещается в инициализирующей части.

2.8.17. Оператор снятия блокировки.

Формат оператора:

```
ДЕБЛОКИРОВАТЬ;
```

Для исключения возможности зависания пользовательской программы в структуру контроллеров серии ЭК-2000 введен охранный таймер - устройство, которое требует обращения к себе через каждые 1,6 секунд. Если такого обращения в течение данного промежутка времени не происходит, охранный таймер генерирует сигнал сброса, приводящий к перезапуску программы. Для обращения к охранному таймеру в программе, написанной на языке CONT, используется оператор ДЕБЛОКИРОВАТЬ. Кроме того, при первом выполнении данного оператора снимается блокировка с выходов контроллера. Блокировка устанавливается для того, чтобы запретить включение устройств, подключенных к выходам модулей вывода, до тех пор, пока порты вывода не будут проинициализированы.

Исходя из назначения оператор ДЕБЛОКИРОВАТЬ, его целесообразно размещать в пользовательской программе после занесения в порты вывода контроллера начальных значений в качестве первого оператора рабочего цикла.

Кроме того, данный оператор следует размещать внутри тех циклов, время выполнения которых заведомо превышает 1,6 секунд (как правило, грамотно спроектированные технологические программы не содержат подобных циклов).

Пример 1:

```
" Инициализирующая часть программы:

ПОРТ_1 = 0;
ПОРТ_2 = FFH;

" Рабочий цикл программы:

[НАЧАЛО_РАБОЧЕГО_ЦИКЛА]
  ДЕБЛОКИРОВАТЬ;
  ".............
ИДТИ на метку (НАЧАЛО_РАБОЧЕГО_ЦИКЛА);
```

Пример 2:

```
"Включить дискретный выход на 1 минуту:
```

```
НАГРЕВАТЕЛЬ = 1;
ОЖИДАТЬ (6000);
НАГРЕВАТЕЛЬ = 0;
```

В примере 2 использование оператора ДЕБЛОКИРОВАТЬ в цикле выдержки времени необходимо (иначе через 1,6 секунд сработал бы охранный таймер). Однако подобное построение технологической программы не является грамотным: время выполнения рабочего цикла возрастает здесь на 1 минуту, что значительно увеличивает время реакции системы.

В следующем примере включение дискретного выхода на 1 минуту организовано другим способом, при этом время выполнения рабочего цикла не меняется:

```
" Инициализирующая часть программы:

ЗАГРУЗИТЬ (ТАЙМЕР) уставкой (6000) в режиме (0);
СТАРТ (ТАЙМЕР);
НАГРЕВАТЕЛЬ = 1;
ПРЕРЫВАНИЕ уровня (16) на подпрограмму (ВЫКЛЮЧЕНИЕ),
если (@ТАЙМЕР@) (==) (1);
РАЗРЕШИТЬ прерывания уровня (16);

" Рабочий цикл программы:

[НАЧАЛО_РАБОЧЕГО_ЦИКЛА]
".....................
ИДТИ на метку (НАЧАЛО_РАБОЧЕГО_ЦИКЛА);

" Подпрограмма обслуживания прерывания уровня 16:

ПОДПРОГРАММА (ВЫКЛЮЧЕНИЕ) {
  НАГРЕВАТЕЛЬ = 0;
}
```

2.8.18. Операторы работы с прерываниями.

Важной особенностью задач управления технологическими процессами является необходимость гибкого переключения задач и оперативного реагирования на возникновение различных ситуаций. Также часто возникает необходимость в параллельном выполнении нескольких задач. Для выполнения этих требований в ЯП CONT включен механизм программного и аппаратного прерывания обычного хода выполнения программ при наступлении некоторого события. Имеются 8 уровней программных и 8 уровней аппаратных прерываний, различающихся по приоритету.

Формат оператора определения программного прерывания:

```
ПРЕРЫВАНИЕ (номер) (операнд_1) (операнд_2) (условие) (операнд_3);
```

номер - номер программного прерывания, задается числовой или символической константой диапазоне 16...23.

операнд_1 - имя подпрограммы обслуживания прерывания.

операнд_2: - регистр,
- флаг,
- таймерный счетчик,

- таймерный бит управления,
- таймерные биты состояния
- порт,
- дискретный вход/выход

условие: == равно (= - в старом формате),
!= не равно (# - в старом формате),
> больше,
< меньше,
>= больше или равно,
<= меньше или равно.

операнд_3: - регистр,
- флаг,
- таймерный счетчик,
- таймерный бит управления,
- таймерные биты состояния,
- порт,
- дискретный вход/выход,
- числовая или символическая константа в диапазоне -32768…32767 (0…FFFFH), если операнд_2 является регистром, в диапазоне 0…255 (0…FFH), если операнд_2 является портом, и в диапазоне 0…1, если операнд_2 является дискретным входом/выходом, таймерным битом или флагом.

Прерывание 16-го уровня имеет высший приоритет, 23-го - низший, т.е. прерывание с меньшим номером может прервать программу обработки прерывания с большим номером. Таким образом могут быть организованы вложенные прерывания. После окончания обработки прерывания продолжается выполнение команд с прерванного адреса.

Для разрешения программного прерывания используется следующий оператор:

РАЗРЕШИТЬ (номер);

номер - номер программного прерывания, задается числовой или символической константой диапазоне 16...23.

Только после выполнения этого оператора операционная система будет проверять выполнение условия, заданного в соответствующем операторе ПРЕРЫВАНИЕ. Проверка производится каждые 10 милисекунд. Если в операторе ПРЕРЫВАНИЕ указано условие, которое выполняется постоянно, то каждые 10 милисекунд будет выполняться соответствующая подпрограмма обслуживания прерывания.

Для запрещения (маскирования) программного прерывания используется следующий оператор:

ЗАПРЕТИТЬ (номер);

номер - номер программного прерывания, задается числовой или символической константой диапазоне 16...23.

Для выделения в программе подпрограммы обслуживания прерывания используется следующий оператор:

ПОДПРОГРАММА (имя_подпрограммы) составной_оператор

Данный оператор нельзя использовать внутри составных операторов. Все подпрограммы обслуживания прерываний должны следовать после рабочего цикла программы.

Одной подпрограмме обслуживания прерывания может соответствовать только один номер прерывания.

Пример использования операторов:

```
" Инициализирующая часть программы:

ЗАГРУЗИТЬ (ТАЙМЕР) уставкой (6000) в режиме (0);
СТАРТ (ТАЙМЕР);
НАГРЕВАТЕЛЬ = 1;
ПРЕРЫВАНИЕ уровня (16) на подпрограмму (ВЫКЛЮЧЕНИЕ),
если (@ТАЙМЕР@) (==) (1);
РАЗРЕШИТЬ прерывания уровня (16);

" Рабочий цикл программы:

[НАЧАЛО_РАБОЧЕГО_ЦИКЛА]
".....................
ИДТИ на метку (НАЧАЛО_РАБОЧЕГО_ЦИКЛА);

" Подпрограмма обслуживания прерывания уровня 16:

ПОДПРОГРАММА (ВЫКЛЮЧЕНИЕ) {
  НАГРЕВАТЕЛЬ = 0;
}
```

В ходе выполнения программы можно многократно разрешать и маскировать программные прерывания, а также переназначать подпрограммы обслуживания и условия прерывания для каждого уровня.

Помимо обслуживания программных прерываний возможно также обслуживание аппаратных прерываний, генерируемых модулями УСО и другими устройствами контроллера. Обработчики аппаратных прерываний называются драйверами. Они представляют собой объектные модули, хранящиеся, как и функции, в библиотеках и присоединяющиеся к коду программы на этапе компоновки. Драйверы, наиболее часто используемые при написании технологических программ, имеются в библиотеках, входящих в состав ИСР ПО CONT-Designer.

Открытый формат оформления драйверов позволяет пользователям ИСР ПО CONT-Designer создавать собственные драйверы для дальнейшего использования их в программе.

Формат оператора инициализации аппаратного прерывания:

```
ДРАЙВЕР (имя_драйвера),
входные  параметры ( вх_пар_1,...,вх_пар_N )
выходные параметры ( вых_пар_1,...,вых_пар_M );
```

```
вх_пар1,...,вх_пар_N: - регистр,
                      - флаг,
                      - таймерный счетчик,
                      - таймерный бит управления,
                      - таймерные биты состояния,
                      - порт,
                      - дискретный вход/выход,
                      - числовая или символическая константа
                        в диапазоне -32768...32767.

вых_пар_1,...,вых_пар_M: - регистр,
                         - флаг,
                         - порт, доступный для записи,
                         - дискретный выход.
```

Оператор ДРАЙВЕР выполняет следующие действия: выделение блока памяти и загрузка в него исполняемого кода драйвера, передачу в стек входных параметров, вызов инициализационной части драйвера, извлечение из стека выходных параметров. После инициализации драйвера обработчик соответствующего аппаратного прерывания остается в памяти контроллера и автоматически выполняется при возникновении прерывания.

Входные (или выходные), или те и другие параметры драйвера могут отсутствовать.

Число входных (или выходных) параметров может быть фиксированным или переменным. Если число входных (или выходных) параметров является фиксировано, то оно должно лежать в диапазоне от 0 до 254.

Если число входных (или выходных) параметров является переменным, то при компиляции отключается проверка соответствия числа входных (или выходных) параметров, указанного в заголовке, и фактического их числа в операторе ДРАЙВЕР. Это позволяет передавать в драйвер переменное число входных и получать переменное число выходных параметров. В этом случае минимальное число входных (или выходных) параметров равно одному, а максимальное общее количество входных и выходных параметров ограничивается размером стека (8 Кбайт).

Для того, чтобы использовать в программе нестандартный драйвер, в инициализирующей части программы должна быть указана библиотека, в которой находится этот драйвер.

Пример использования оператора:

```
" Инициализация стандарного драйвера для работы с часами
" реального времени из библиотеки LIB\SYSTEM.LIB
ДРАЙВЕР (D3RTC) входной параметр (#ТЕК_СЕК);
" Разрешение прерываний от часов реального времени
ФУНКЦИЯ (FINTRTC) входной параметр (1);
```

При использовании в программе драйверов, обслуживающих УСО, необходимо разрешить прерывания от соответствующих УСО. Для этого используется следующий оператор:

```
РАЗРЕШИТЬ (номер_слота);
```

номер_слота - числовая или символическая константа в диапазоне 0...6.

Для запрещения (маскирования) прерывания от заданного УСО используется следующий оператор:

```
ЗАПРЕТИТЬ (номер_слота);
```

2.9. Псевдооператоры.

В отличие от операторов псевдооператоры не генерируют объектный код и предназначены для управления процессами трансляции и компоновки программы. В ЯП CONT используется только один псевдооператор - БИБЛИОТЕКА.

Псевдооператор БИБЛИОТЕКА предназначен для указания компилятору и компоновщику расположения на диске и имени библиотеки, содержащей используемые в программе функции и драйверы. В результате компилятору становится доступной информация о параметрах функций и

драйверов, а компоновщику - о местонахождении соответствующих объектных модулей.

Стандартные библиотеки MATH.LIB, MEM.LIB, MODULE.LIB и SYSTEM.LIB подключаются к проекту автоматически, поэтому их можно не объявлять в программе.

Количество используемых в программе библиотек может достигать 50. Данные псевдооператоры размещаются в инициализирующей части программы. Формат псевдооператора БИБЛИОТЕКА:

```
БИБЛИОТЕКА (путь:имя_библиотеки);
```

2.10. Структура программ.

Любая программа, написанная на ЯП CONT, состоит из последовательности логических частей (подзадач), каждая из которых имеет в программе свое определенное назначение. Условно такими логическими частями программы являются:

- инициализирующая часть;
- рабочий цикл;
- процедуры;
- подпрограммы обслуживания прерываний.

Инициализирующая часть представляет собой последовательность операторов и псевдооператоров, выполняющихся однократно в начале работы программы. Здесь, как правило, выполняются следующие действия:

- определяются библиотеки функций и драйверов, используемых в программе;
- портам, дискретным выходам, регистрам и флагам присваиваются начальные значения; таймеры загружаются уставками;
- задаются программные и аппаратные прерывания;
- в операторе РЕСТАРТ определяется метка, на которую передается управление в случае горячего перезапуска контроллера.

Рабочий цикл представляет собой последовательность операторов, выполняющихся циклически. Здесь размещаются операторы, вызываются процедуры и функции, обеспечивающие чтение входных параметров и правление технологическим процессом. В рабочем цикле также осуществляется обращение к охранному таймеру контроллера.

Процедуры представляют собой логически связанные последовательности операторов, которые могут быть многократно вызваны для выполнения из различных точек программы.

Подпрограммы обслуживания прерывания представляет собой последовательности операторов, выполняющихся каждый раз, когда возникают определенные события, заданные в инициализирующей части. При этом выполнение рабочего цикла программы прерывается.

Примером, в котором используются все перечисленные логические части, может быть программа PID_IMP.CON, содержащаяся в каталоге `..\WinCont\Examples\Programs`.

2.11. Использование модульного программирования.

При создании достаточно крупных проектов целесообразно составлять программу из нескольких программных модулей. Такой метод, названный модульным программированием, при котором логически связанные процедуры и подпрограммы оформляются в отдельные модули,

позволяет использовать весь объем памяти пользовательских программ контроллера (до 512 Кбайт), упростить чтение, отладку, модификацию программы и сократить время ее компиляции.

Память пользовательских программ разбита на сегменты размером по 32 Кбайт, в каждом из которых может размещаться только один модуль. Таким образом, пользовательская программа может состоять не более чем из 16 модулей.

Один из программных модулей должен быть главным; в нем размещаются инициализирующая часть программы и основной рабочий цикл. Главный модуль может содержать также процедуры и подпрограммы обслуживания прерываний. Файл, содержащий исходный текст главного модуля, должен иметь расширение .CON.

Остальные модули являются дополнительными; в каждом из них размещаются одна или несколько логически связанных между собой процедур и подпрограмм обслуживания прерываний. Файлы, содержащие исходные тексты дополнительных модулей, должны иметь расширения .Cxx (xx - номера модулей).

Модуль с номером 1 считается самым младшим, главный модуль - самым старшим. Процедуры, расположенные в том или ином модуле, могут быть вызваны из этого или более старшего модуля. Аналогично, подпрограммы обслуживания прерываний, расположенные в том или ином модуле, могут быть определены в командах ПРЕРЫВАНИЕ, находящихся в этом или в более старших модулях. Таким образом, процедуры и подпрограммы, расположенные в самом младшем модуле, могут использоваться во всех модулях. Процедуры и подпрограммы, находящиеся в главном модуле, могут использоваться только в этом модуле.

Модульное программирование позволяет также использовать метод групповой разработки, при котором написанием программных модулей занимаются несколько программистов.

2.12. Использование вещественных чисел.

В ЯП CONT имеется возможность работы с 32 - разрядными числами, заданными в формате с плавающей точкой (вещественными числами), которые физически располагаются в двух соседних регистрах (или элементах массива регистров). В регистре (элементе массива регистров) с меньшим порядковым номером (индексом) располагаются младшие 16 разрядов вещественного числа. В регистре (элементе массива регистров) со следующим по счету порядковым номером (индексом) располагаются старшие 16 разрядов вещественного числа.

В режиме отладки возможно просматривать вещественное значение в окне переменных или в окне слежения, а также модифицировать это значение. При этом необходимо использовать имя регистра с меньшим порядковым номером и выбирать вещественную форму представления «Real» («rl»). Если вещественное значение превышает верхний допустимый диапазон (±3.4E+38), то вместо значения отображается слово «Переполнение».

Пример: вещественное число находится в регистрах РЕЗУЛЬТАТ_МЛ с порядковым номером 50 и РЕЗУЛЬТАТ_СТ с порядковым номером 51. Для просмотра вещественного числа в окне переменных, его изменения или занесения в окно слежения необходимо выбрать регистр РЕЗУЛЬТАТ_МЛ и вещественную форму представления «Real» («rl»).

Для записи вещественных чисел в регистры и выполнения арифметических действий над такими числами (сложения, вычитания, умножения, деления, вычисления натурального логарифма, возведения в степень, извлечения квадратного корня, выделения мантиссы и порядка,

масштабирования) используются функции библиотеки MATH.LIB.

Ниже приводится фрагмент программы, написанной на ЯП CONT, иллюстрирующий работу с вещественными числами. Допустим, что в окне данных заданы следующие регистры:

ЧИСЛО1_МЛ - младшие разряды первого слагаемого;
ЧИСЛО1_СТ - старшие разряды первого слагаемого;
ЧИСЛО2_МЛ - младшие разряды второго слагаемого;
ЧИСЛО2_СТ - старшие разряды второго слагаемого;
РЕЗУЛЬТАТ_МЛ - младшие разряды результата;
РЕЗУЛЬТАТ_СТ - старшие разряды результата;
МАНТИССА - мантисса результата;
ПОРЯДОК - порядок результата;
ОШИБКА - код ошибки выполнения операций над вещественными числами (переполнения, деления на нуль, неправильно заданных входных параметров и т.д.).

Фрагмент программы:

```
" Образование первого вещественного слагаемого (12.55)
" из целочисленных мантиссы (1255) и порядка (-2):

ФУНКЦИЯ (FTOE)
входные данные ( 1255, -2 )
выходные данные ( ЧИСЛО1_МЛ, ЧИСЛО1_СТ, ОШИБКА );

ЕСЛИ (ОШИБКА != 0), то {
  ВЫЗВАТЬ процедуру (ОБРАБОТКА_ОШИБКИ);
}

" Образование второго вещественного слагаемого (92.45)
" из целочисленных мантиссы (9245) и порядка (-2):

ФУНКЦИЯ (FTOE)
входные данные ( 9245, -2 )
выходные данные ( ЧИСЛО2_МЛ, ЧИСЛО2_СТ, ОШИБКА );

ЕСЛИ (ОШИБКА != 0), то {
  ВЫЗВАТЬ процедуру (ОБРАБОТКА_ОШИБКИ);
}

" Сложение полученных вещественных слагаемых:

ФУНКЦИЯ (ADDF)
входные данные ( ЧИСЛО1_МЛ, ЧИСЛО1_СТ, ЧИСЛО2_МЛ, ЧИСЛО2_СТ )
выходные данные ( РЕЗУЛЬТАТ_МЛ, РЕЗУЛЬТАТ_СТ, ОШИБКА );

ЕСЛИ (ОШИБКА != 0), то {
  ВЫЗВАТЬ процедуру (ОБРАБОТКА_ОШИБКИ);
}

" Выделение из вещественного результата сложения
" целочисленных мантиссы и порядка:

ФУНКЦИЯ (ETOF)
входные данные ( РЕЗУЛЬТАТ_МЛ, РЕЗУЛЬТАТ_СТ )
выходные данные ( МАНТИССА, ПОРЯДОК );

".....................................................

" Процедура обработки ошибки выполнения
```

```
" операций над вещественными числами:

ПРОЦЕДУРА (ОБРАБОТКА_ОШИБКИ) {
  "..................................................
   ОШИБКА = 0;
}
```

2.13. Массивы переменных и символических констант.

В языке CONT имеется возможность работы с массивами переменных (регистров и таймеров) и символических констант. Присвоение порядковых номеров регистрам и таймерам позволяет организовывать массивы переменных. Например, если за регистром НАЧАЛО_АРХИВА_1 с номером 100, мы назначаем регистр НАЧАЛО_АРХИВА_2 с номером 200, то тем самым мы определяем массив размером в 100 регистров. Массивом символических констант считается заданный в окне данных список следующих друг за другом констант.

Для обращении к элементу массива переменных после имени первого элемента массива в квадратных скобках указывается смещение адресуемого элемента относительно начала массива. Смещением может быть или непосредственно номер элемента, представленный числом или символической константой, или имя регистра-индекса. В последнем случае текущее значение регистра-индекса будет являться номером адресуемого элемента относительно начала массива.

Пример 1:

```
ЗАГРУЗИТЬ таймер ( ВЫДЕРЖКА_ВРЕМЕНИ[НОМЕР_ЗАДВИЖКИ] )
уставкой ( ВРЕМЯ_ХОДА[НОМЕР_ЗАДВИЖКИ] ), режим (0);
```

В данном примере таймер из массива ВЫДЕРЖКА_ВРЕМЕНИ загружается уставкой из массива регистров ВРЕМЯ_ХОДА. Номера элементов указанных массивов равны текущему значению регистра НОМЕР_ЗАДВИЖКИ, в котором содержится номер управляемой задвижки.

Пример 2:

```
@ВЫДЕРЖКА_ВРЕМЕНИ[ЗАДВИЖКА_1] = ! бит состояния ТАЙМЕР@[1] &
                                 ! бит состояния ТАЙМЕР@[2];
```

Здесь таймер из массива ВЫДЕРЖКА_ВРЕМЕНИ (номер элемента массива выражен символической константой ЗАДВИЖКА_1) запускается, если таймеры с номерами 1 и 2 из массива ТАЙМЕР досчитали до 0, т.е. их биты состояния имеют значение "нуль".

Для работы с массивами служат также следующие функции, которые содержащиеся в библиотеке LIB\MEM.LIB:

CONCPY - копирование массива символических констант
в массив регистров.
RD_CON - чтение элемента массива символических констант.
RD_FLAG - чтение разряда элемента массива регистров.
RD_BIT - чтение элемента массива флагов.
REGCPY - копирование массива регистров.
REGSET - заполнение массива регистров значением.
WR_FLAG - запись в разряд элемента массива регистров.
WR_BIT – запись в элемент массива флагов.

Описание перечисленных функций и примеры их использования содержатся в файле LIB\MEM.DOC. При обращении к элементу массива переменных или символических констант в перечисленных функциях указываются порядковый номер первого элемента массива и номер элемента в массиве (т.е. индекс). Порядковым номером символической константы считается ее номер в списке заданных в окне данных символических констант. Для получения порядкового номера переменной или символической константы в программе необходимо указать перед именем унарную операцию взятия номера "#".

Заметим, что при написании функций имеется возможность использования механизма выделения и освобождения памяти данных для организации массивов в дополнительном сегменте памяти данных контроллера. Однако доступ к элементам таких массивов из SCADA-систем будет невозможен.

3. Написание функций и драйверов.

Открытый формат оформления функций и драйверов позволяет пользователям ИСР ПО CONT-Designer самостоятельно расширять ее возможности путем написания собственных функций и драйверов для дальнейшего использования их в программе.

Как функции, так и драйверы пишутся на языке ассемблера для микропроцессора Intel 80186. Формат оформления функций позволяет осуществлять их отладку путем их вызова из программ, написанных, например, в среде Borland C++. Для создания, просмотра и модификации библиотек в ИСР ПО CONT-Designer включена специальная сервисная программа TCONTLIB.EXE.

В папке LIB содержатся библиотеки SYS03B.LIB, SYS11A.LIB, SYS17A.LIB, SYS17B.LIB, позволяющие создавать аппаратно-зависимые функции и драйверы с одинаковыми именами. При использовании таких функций и драйверов модификация прикладных программ при переходе на другой тип центрального процессорного модуля не требуется. В зависимости от типа центрального процессорного модуля при компоновке прикладной программы к ней автоматически присоединяется та или иная библиотека.

3.1. Создание функций.

При написании функции в ее заголовке указывается число входных и выходных параметров. Число входных, а также число выходных параметров функции может лежать в диапазоне от 0 до 0FEh (254).

Если в заголовке функции указано число входных или выходных параметров, равное 0FFh (255), то при компиляции отключается проверка соответствия числа входных или выходных параметров, указанного в заголовке, и фактического их числа в команде ФУНКЦИЯ. Это позволяет передавать в функцию переменное число входных и получать переменное число выходных параметров. В этом случае минимальное число параметров равно одному. В случае переменного числа параметров функции, максимальное общее количество входных и выходных параметров ограничивается размером стека (8 Кбайт).

Процесс создания функции включает следующие этапы:

1) Написание текста функции на языке ассемблера в каком-либо текстовом редакторе. Имя ассемблерного файла обязательно должно совпадать с именем функции.

Исходный текст функции должен удовлетворять определенным требованиям. Функция должна:

- иметь заголовок,
- настроиться на область аргументов функции,
- сохранить в стеке используемые регистры процессора,
- восстановить использованные регистры процессора из стека при завершении работы.

Ниже приводится пример оформления функции. Данная функция возвращает сумму двух входных параметров:

```
.186
.MODEL  SMALL
 public _add
.CODE

;       Заголовок функции

      db    'HEADER'
      db    2     ;Число входных параметров
      db    1     ;Число выходных параметров

;       Начало функции

_add    proc  near
ARG     val1, val2 RETURNS result

;       _add       - имя функции
;       val1,val2  - входные параметры
;       result     - выходной параметр

;       Настройка на область аргументов функции

      push  bp
      mov   bp,sp

;       Сохранение используемых регистров

      push  ax

;       Текст функции

      mov   ax,val1
      add   ax,val2
      mov   result,ax

;       Восстановление использованных регистров

      pop   ax
      pop   bp

;       Выход из функции

      ret
_add  endp
      end
```

Следующий пример является модификацией предыдущего и демонстрирует написание функции с переменным числом параметров. Данная функция возвращает сумму переменного числа входных параметров (слагаемых). Фактическим числом слагаемых здесь является значение первого входного параметра:

```
.186
.MODEL  SMALL
 public _addnew
.CODE

StackStruct   struc     ; Структура стека
SaveBP  dw    ?         ; Копия регистра BP
RetAddr dw    ?         ; Адрес возврата
;       Зона входных параметров:
ParNum  dw    ?         ; Число слагаемых
Input1  dw    ?         ; 1-е слагаемое
;       Далее следуют остальные слагаемые
;       Затем следует зона выходных параметров
StackStruct   ends

;       Заголовок функции

      db    'HEADER'
      db    0FFh      ;Число входных параметров является переменным
      db    1         ;Число выходных параметров

;       Начало функции

_addnew proc  near

;       Настройка на область аргументов

      push  bp
      mov   bp,sp

;       Сохранение используемых регистров

      push  ax
      push  si
      push  cx

;       Текст функции

      xor   ax,ax             ;обнулить результат
      mov   si,ax             ;и номер слагаемого
      mov   cx,[bp+ParNum]    ;поместить число слагаемых в CX
      or    cx,cx             ;проверить CX на нуль
      jz    Quit
L:
      add   ax,[bp+Input1+si] ;в регистре SI-смещение текущего
      add   si,2              ;слагаемого относительно первого
      loop  L
Quit:
;       В стеке после последнего входного параметра следует
;       зона выходных параметров. В данный момент регистр
;       SI указывает на первый выходной параметр.

      mov   [bp+Input1+si],ax ;запомнить результат
```

```
;       Восстановление использованных регистров

      pop   cx
      pop   si
      pop   ax
      pop   bp

;       Выход из функции

      ret
_addnew endp
      end
```

2) Компиляция функции программой TASM.EXE или MASM.EXE.

Заметим, что если в первом примере для описания параметров функции использовалась конструкция ARG ... RETURNS ..., обрабатываемая только программой TASM.EXE, то в последнем примере для этой цели используется структура StackStruct. Описание параметров с помощью подобной структуры позволяет компилировать функцию как программой TASM.EXE, так и программой MASM.EXE.

3) Включение полученного объектного модуля в библиотеку объектных модулей с помощью программы TCONTLIB.EXE.

В каталоге EXAMPLES\FUNC&DRV содержится следующая полезная информация:

- шаблон оформления функции FUNC_EX.ASM;
- тексты приведенных выше примеров функций;
- программы, написанные в среде BORLAND C++, вызывающие эти функции при их отладке.

При написании функций имеется возможность выделения памяти данных для размещения в нем вспомогательных переменных и массивов переменных. Блок памяти выделяется в дополнительном сегменте памяти данных контроллера.

При написании функций, в которых производятся арифметические вычисления, после команд, модифицирующих микропроцессорный флаг переполнения OF, целесообразно использовать команду прерывания по переполнению INTO. Данная команда в случае возникновения переполнения инициирует прерывание типа 4. Обработчик этого прерывания, находящийся в операционной системе OS-188, взводит системный флаг ПЕРЕПОЛНЕНИЕ, находящийся в регистре STATUS.

Если в функции имеется обращение к ячейкам сегмента кода CS, то необходимо создать второй экземпляр этой функции, который также необходимо откомпилировать и включить в ту же библиотеку, в которую была помещена исходная функция. Второй экземпляр отличается от первого следующим:

- имя файла, содержащего исходный текст функции, должен состоять из имени функции, перед которым ставится символ ‘$’;

- во всех местах исходного текста, в которых указывается имя функции, перед именем необходимо поставить символ ‘$’;

- во всех местах исходного текста, в которых имеется обращение к ячейкам сегмента кода, (например, mov ax, cs:[bx]), необходимо в качестве смещения добавить к адресу значение 8000H, (например, mov ax, cs:[bx+8000H]).

3.2. Создание драйверов.

Драйвер состоит из двух частей - инициализационной части и обработчика прерывания (резидентной части). После использования команды ДРАЙВЕР управление передается инициализационной части драйвера, в которой производятся следующие действия: извлечение из стека входных параметров и, при необходимости, их сохранение в памяти драйвера, обращение к системной функции для настройки соответствующего вектора на свой обработчик прерывания, помещение в стек выходных параметров и выход.

Входные параметры, сохраненные инициализационной частью в памяти драйвера, могут быть использованы в обработчике прерывания.

Обработчик прерывания драйвера начинает работать при возникновении разрешенного аппаратного прерывания от соответствующего слота. Обработчик осуществляет следующие действия: обработку возникшего прерывания (эти действия являются специфическими для конкретного драйвера), обращение к системной функции с запросом завершения прерывания и выход.

В каталоге EXAMPLES\FUNC&DRV содержится шаблон оформления драйверов. В заголовке драйвера указывается число входных и выходных параметров. Число входных, а также число выходных параметров драйвера может лежать в диапазоне от 0 до 0FEh (254).

Если в заголовке драйвера указано число входных или выходных параметров, равное 0FFh (255), то при компиляции отключается проверка соответствия числа входных или выходных параметров, указанного в заголовке, и фактического их числа в команде ДРАЙВЕР. Это позволяет передавать в драйвер переменное число входных и получать на выходе его инициализационной части переменное число выходных параметров. В этом случае минимальное число параметров равно одному. В случае переменного числа параметров драйвера, максимальное общее количество входных и выходных параметров ограничивается размером стека (8 Кбайт).

Процесс создания драйвера включает следующие этапы:

1) Написание текста драйвера на языке ассемблера в каком-либо текстовом редакторе. Имя ассемблерного файла обязательно должно совпадать с именем драйвера.

Исходный текст драйвера должен удовлетворять определенным требованиям. Инициализационная часть драйвера должна:

- иметь заголовок,
- зарезервировать слова, соответствующие входным параметрам драйвера, используемым в резидентной части,
- настроиться на область аргументов,
- сохранить в стеке используемые регистры процессора,
- сохранить значения входных параметров драйвера, используемых в резидентной части,
- установить вектор прерывания,
- инициализировать устройства, используемые в резидентной части,
- восстановить использованные регистры процессора.

Резидентная часть драйвера должна:

- сохранить в стеке флаги и используемые регистры процессора,
- выполнить обработку возникшего аппаратного прерывания,
- завершить прерывание (если драйвер обслуживает слот),
- восстановить флаги и использованные регистры процессора.

2) Компиляция драйвера программой TASM.EXE или MASM.EXE.

Заметим, что конструкция ARG ... RETURNS ..., в которой перечисляются входные и выходные параметры, обрабатывается только программой TASM.EXE. Описание параметров с использованием структуры стека позволяет компилировать драйвер как программой TASM.EXE, так и программой MASM.EXE.

3) Включение полученного объектного модуля в библиотеку объектных модулей с помощью программы TCONTLIB.EXE.

При написании драйверов имеется возможность выделения памяти данных для размещения в нем вспомогательных переменных и массивов переменных. Блок памяти выделяется в дополнительном сегменте памяти данных контроллера.

Если в драйвере имеется обращение к ячейкам сегмента кода CS, то необходимо создать второй экземпляр этого драйвера, который также необходимо откомпилировать и включить в ту же библиотеку, в которую был помещен исходный драйвер. Второй экземпляр отличается от первого следующим:

- имя файла, содержащего исходный текст драйвера, должен состоять из имени драйвера, перед которым ставится символ ‘$’;

- во всех местах исходного текста, в которых указывается имя драйвера, перед именем необходимо поставить символ ‘$’;

- во всех местах исходного текста, в которых имеется обращение к ячейкам сегмента кода, (например, mov ax, cs:[bx]), необходимо в качестве смещения добавить к адресу значение 8000H, (например, mov ax, cs:[bx+8000H]).

3.3. Использование программы TCONTLIB.EXE.

Для работы с библиотеками функций и драйверов используется программа TCONTLIB.EXE, находящаяся в каталоге LIB.

Формат командной строки программы TCONTLIB.EXE следующий:

```
TCONTLIB <библиотека> [ключ1] [мод1]...[ключN] [модN]
или:
TCONTLIB <библиотека> ?
```

Пример 1: TCONTLIB LIB\MYLIB +- MY_DRV + MY_FUNC
Пример 2: TCONTLIB LIB\MATH ?

Здесь <библиотека> - имя библиотеки без расширения; перед именем библиотеки может быть указан путь к ней;

[мод1]...[мод2] - имена объектных модулей без расширений, перед которыми могут стоять пути;

Ключи [ключ1]...[ключN] могут быть следующие:

+	добавление модуля в библиотеку;
-	удаление модуля из библиотеки;
*	извлечение модуля из библиотеки;
+- или -+	обновление модуля в библиотеке;
- или -	извлечение модуля и удаление его из библиотеки.

Второй вариант вызова программы TCONTLIB.EXE обеспечивает вы-

вод на экран содержимого библиотеки с указанием имен функций, их размеров, а также числа входных и выходных параметров для каждой функции.

Командная строка для создания библиотеки выглядит так же, как для добавления модуля (модулей) в эту библиотеку.

При запуске программы без параметров на экран выдается подсказка о ключах программы.

3.4. Механизм выделения и освобождения памяти.

При написании функций и драйверов имеется возможность выделения блока памяти данных для размещения в нем вспомогательных переменных и массивов переменных. Блок памяти выделяется в дополнительном сегменте памяти данных контроллера.

Для выделения блока памяти необходимо включить в текст функции или драйвера следующую последовательность команд:

```
mov   bx, число_параграфов_памяти
mov   ah, 1
int   4Bh
```

После выделения памяти необходимо проанализировать флаг переноса процессора. В случае успешного выделения памяти флаг переноса сбрасывается, а в регистр AX заносится начальный адрес выделенного блока памяти.

В случае возникновения ошибки флаг переноса взводится, а в регистр AX заносится код ошибки. Возможными кодами ошибки являются 07h (уничтожены блоки управления памятью) и 08h (нехватка памяти). Если для удовлетворения запроса не хватает памяти, в регистре BX будет сообщен размер (в параграфах) самого большого из имеющихся блоков памяти.

Для освобождения блока памяти необходимо включить в функцию следующую последовательность команд:

```
mov   es, адрес_начального_параграфа_освобождаемого_блока
mov   ah, 2
int   48h
```

После выделения памяти необходимо проанализировать флаг переноса процессора. В случае успешной очистки памяти флаг переноса сбрасывается.

В случае возникновения ошибки флаг переноса взводится, а в регистр AX заносится код ошибки. Возможными кодами ошибки являются: 06h (некорректно задан адрес блока памятью) и 07h (уничтожены блоки управления памятью).

4. Установка и подготовка к работе.

Для работы с интегрированной системой разработки (ИСР) прикладного программного обеспечения (ПО) CONT-Designer необходимо установить этот программный продукт с поставляемых дискет (или CD-ROM) на жесткий диск Вашего компьютера.

Требования к компьютеру: IBM PC AT 386 или выше, размер ОЗУ - 4 Мбайт или более. Операционная система: Windows 98/Me или Windows NT 4.0/2000/XP.

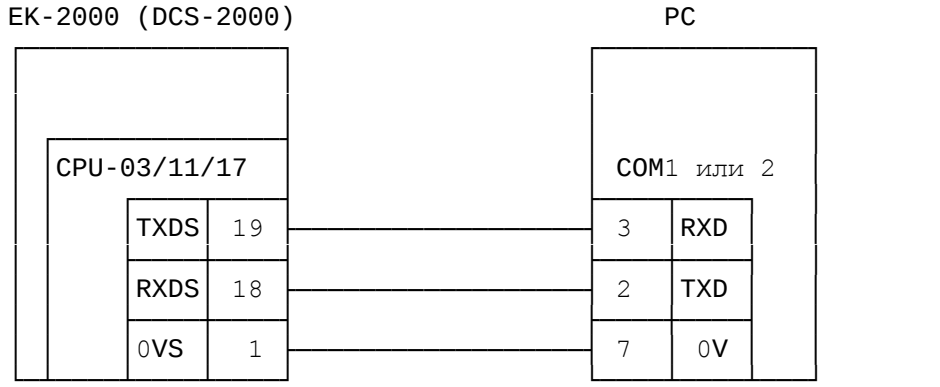
Если на Вашем компьютере уже установлена ИСР ПО CONT-Designer for Windows предыдущей версии, ее необходимо удалить, воспользовавшись стандартным диалогом Windows «Пуск | Настройка | Панель управления | Установка и удаление программ».

Для установки программного пакета следует запустить файл SETUP.EXE, находящийся на диске (или дискете #1) поставляемого дистрибутива, или воспользоваться стандартным диалогом Windows «Пуск | Настройка | Панель управления | Установка и удаление программ». Далее необходимо следовать инструкциям программы установки, по окончании которой Вам будет предложено перезагрузить компьютер. Если устанавливается новая версия программного пакета вместо более ранней версии, перезагрузка компьютера не требуется.

После установки создается новый каталог (по умолчанию - ..\Program Files\Emicon\WinCont), содержащий файлы ИСР ПО CONT-Designer. В меню Windows «Пуск | Программы» создается новая программная группа «CONT-Designer».

Для отладки программ необходимо подключить контроллер к одному из последовательных портов компьютера, либо к сетевому модулю, установленному в компьютер.

Загрузка программы в контроллер и обмен данными между контроллером и компьютером при отладке может осуществляться через системный или несистемный последовательный канал RS-232, либо через сетевой модуль (интерфейс RS-485). Ниже приведена схема подключения через системный канал RS-232:



(для разъема на 25 контактов)

5. Интегрированная среда.

Ключевым в ИСР ПО CONT-Designer является понятие проекта.

Проект в CONT-Designer – это совокупность составляющих его программных модулей, а также информация об аппаратной конфигурации контроллера, сетевых настройках и используемых в программе данных. Далее по тексту проект будет также называться программой.

5.1. Начало работы.

Из меню Windows «Пуск|Программы|CONT-Designer» запустите программу CONT-Designer. На экране появляется главное окно, в верхней части которого находится главное меню. Под главным меню находится панель инструментов с кнопками быстрого доступа к основным операциям. В нижней части окна находится строка состояния, в которой отображаются: подсказки при указании мышью на ту или иную кнопку панели инструментов либо элемент меню; текущие настройки соединения с контроллером; режим работы интегрированной среды (редактирование или просмотр текста); позиция курсора в текстовом редакторе. Для начала работы откройте существующий проект, или создайте новый проект.

Для открытия уже существующего проекта воспользуйтесь командой меню «Проект | Открыть» или кнопкой «Открыть проект» на панели инструментов, или клавишами Ctrl-O. При этом открывается диалоговое окно, в котором предлагается указать имя и расположения файла проекта. Файл проекта имеет расширение .cpr. При первом открытии проекта, созданного в CONT-Designer версии 4.04 или ниже, необходимо выбрать файл с расширением .cnf (в дальнейшем, после первого сохранения проекта, необходимо открывать файл с расширением .cpr).

После открытия проекта становятся доступными ряд операций: открытие составляющих его программных модулей (выполняется путем выбора команды меню «Проект | Выбрать модуль…» или нажатия клавиш Ctrl-M, или соответствующей кнопки на панели инструментов), задание аппаратной конфигурации контроллера (команда меню «Проект | Конфигурация…» или клавиши Ctrl-F12), просмотр и изменение используемых в программе данных (команда меню «Данные | Показать», клавиши Shift-F2 или кнопка «Окно данных» на панели инструментов).

Для создания нового проекта воспользуйтесь командой меню «Проект | Создать». При этом автоматически создается проект с именем Noname.cpr, содержащий программный модуль Noname.con, конфигурацию контроллера по умолчанию и два системных регистра – STATUS и RUNTIMEERROR. При последующем сохранении нового проекта (команда меню «Проект | Сохранить» или «Проект | Сохранить как…») на экране появляется диалоговое окно, в котором необходимо указать имя и расположение сохраняемого проекта.

Также имеется возможность автоматического создания проекта для контроллеров серии DCS-2000, в котором реализуются: обмен данными с модулями DCS-2000 по одному или двум каналам, считывание аналоговых и дискретных входных сигналов, а также положительных и отрицательных фронтов на дискретных входах, упаковка входных сигналов, формирование импульсных и статических сигналов, а также меандров на дискретных выходах, запись аналоговых выходных сигналов, формирование информации о качестве связи с модулями DCS-2000. При этом участок программы, реализующий перечисленные действия, скрыт

от пользователя. Для создания такого проекта используются команды меню «Проект | Создать… | Проект для CPU-11/15», «Проект | Создать… | Проект для CPU-17A» или «Проект | Создать… | Проект для CPU-17B». Далее следует задать используемые модуля связи с объектом серии DCS-2000, при необходимости, изменить сетевые настройки контроллера.

Имена всех ранее открытых проектов запоминаются и отображаются в нижней части меню «Проект» для удобства их повторного открытия.

5.2. Задание аппаратной конфигурации контроллера.

Для задания аппаратной конфигурации контроллера необходимо воспользоваться командой меню «Проект | Конфигурация…» или клавишами Ctrl-F12. Далее на экране появляется окно, содержащее две вкладки: «Модули связи с объектом» и «Сеть», в которых задаются соответственно модули связи с объектом и сетевые настройки контроллера. На вкладке «Сеть» имеется также возможность настроить модули С-02, если они были заданы для контроллера серии ЭК-2000. Изменение аппаратной конфигурации контроллера доступно только в режиме редактирования программы.

На вкладке «Модули связи с объектом» задаются тип центрального процессорного модуля контроллера и тип установленной в него микросхемы флэш-памяти: 2F8010 (128 Кбайт), 2F8020 (256 Кбайт), 29C020 (256 Кбайт), 29C040 (512 Кбайт) или 29F040 (512 Кбайт). В зависимости от размера флэш-памяти максимальное количество программных модулей в проекте может составлять соответственно 4, 8 или 16. Здесь также задаются типы модулей связи с объектом, установленных в слоты контроллера (для серии ЭК-2000), или подключенных к центральному процессорному модулю (для серии DCS-2000).

Для выхода из окна «Конфигурация…» с сохранением измененной аппаратной конфигурации контроллера следует нажать клавишу «OK», а для выхода без сохранения – клавишу «Отмена».

5.2.1. Задание модулей связи с объектом.

Для контроллеров серии ЭК-2000 модули связи с объектом выбираются в пронумерованных полях, связанных со слотами контроллера. В модулях дискретного ввода-вывода можно задать порты ввода-вывода дискретных сигналов. Для этого следует дважды щелкнуть мышью на названии выбранного модуля или нажать клавишу Enter и в полях ввода появившегося окна задать имена портов. Слева от полей ввода указаны номера портов. В портах ввода-вывода можно задать отдельные дискретные входы/выходы (биты). Для этого следует дважды щелкнуть мышью на имени выбранного порта или нажать клавишу Enter и в полях ввода появившегося окна задать имена битов. Слева от полей ввода указаны номера битов.

При удалении из слота модуля, содержащего порты, на экран выводится предупреждающее сообщение с запросом подтверждения удаления этого модуля.

Сетевые модули С-02 следует задавать только в слотах 0…3. Они должны следовать друг за другом, причем в первых слотах должны рас-

полагаться модули С-02, работающие в режиме «Ведомый», а за ними - модули, работающие в режиме «Ведущий».

При перемещении модуля из одного слота в другой необходимо подвести курсор мыши к названию перемещаемого модуля, нажать левую кнопку мыши, не отпуская ее, подвести курсор мыши к требуемому слоту, и отпустить кнопку мыши (в дальнейшем для краткости подобную операцию будем называть «перетаскиванием мышью»). При перетаскивании модуля дискретного ввода-вывода перемещаются все порты и дискретные входы-выходы, назначенные в этих портах. При перетаскивании модуля С-02 перемещается и назначенная в нем карта заявок.

Для контроллеров серии DCS-2000, если проект был создан как «Проект для CPU-11/15», «Проект для CPU-17A» или «Проект для CPU-17B», в окне «Модули связи с объектом» задаются модули, подключенные к выбранным каналам центрального процессорного модуля. При щелчке правой кнопки мыши в этом окне появляется контекстное меню, в котором перечислены возможные операции: добавление нового модуля, изменение адреса, типа или параметров сигналов выбранного модуля, удаление выделенных модулей, перемещение их в другие адреса, копирование выделенных модулей в буфер обмена, вставка модулей из буфера обмена, поиск модуля по заданному адресу или типу, а также сохранение конфигурации модулей.

При добавлении нового модуля серии DCS-2000 появляется окно «Новый модуль…», в котором указываются адрес, тип модуля и каналы центрального процессорного модуля, к которым подключен добавляемый модуль. **Внимание! К каждому информационному каналу центрального процессорного модуля можно подключать не более 32 модулей!**

Для модулей дискретного ввода и ввода-вывода дополнительно указываются типы считываемых дискретных сигналов. С непомеченных входов снимаются значения сигналов DI.

С входов, помеченных как «Положительные фронты», считываются следующие сигналы:

I = PP | DI, где PP – сигнал обнаружения положительного фронта длительностью 2.5 с, считанный с модуля дискретного ввода или ввода-вывода; DI – сигнал, считанный с дискретного входа.

С входов, помеченных как «Отрицательные фронты», считываются следующие сигналы:

I = !NP & DI, где NP – сигнал обнаружения отрицательного фронта длительностью 2.5 с, считанный с модуля дискретного ввода или ввода-вывода; DI – сигнал, считанный с дискретного входа.

Для модулей дискретного вывода и ввода-вывода также указываются типы выходных дискретных сигналов: на выходах, помеченных как «Импульсы на выходах», формируются импульсные сигналы; на выходах, помеченных как «Меандры на выходах», формируются меандры. Длительности импульсов задаются в полях «Длительности импульсов». На непомеченных выходах формируются статические сигналы.

Для просмотра информации об используемых в проекте переменных и константах следует воспользоваться командой меню «Проект | Информация…» и нажать кнопку «Комментарий…».

5.2.2. Задание сетевых настроек контроллера.

По умолчанию операционная система центральных процессорных модулей настраивает информационные каналы на режим «Ведомый» и определенные параметры обмена данными, а также присваивает им сетевые адреса «1».

На вкладке «Сеть» можно изменить сетевые настройки контроллера: для модулей CPU-03/11/15 назначить новый сетевой адрес контроллера, для модулей CPU-17A и CPU-17B назначить сетевой адрес каждому информационному каналу, настроить один или несколько каналов на режим «Ведущий» или назначить новые параметры обмена данными (скорость и т.д.). **Внимание! Задание сетевых адресов каналов в модулях CPU-17A с прошивками версии CP1707 и ниже не поддерживается!** Также можно изменить имя драйвера, обеспечивающего обмен данными по каналам, настроенным на режим «Ведущий». В результате изменений, если в нижней части окна присутствует и установлен флажок «Автоматически добавить процедуру конфигурации в программу», после нажатия на клавишу «OK», сформируется файл с именем, совпадающим с именем проекта и расширением «.NET». В этом файле будет содержаться процедура, которая инициализирует драйвер и вызывает функции, настраивающие каналы. Данная процедура будет автоматически вызвана в начале работы программы.

Новые значения сетевых адресов (для модулей CPU-11/15, CPU-17A и CPU-17B), режимы работы и настройки каналов вступают в силу после дальнейшей загрузки программы в память контроллера и ее запуска.

Назначение сетевого адреса модулю CPU-03 на вкладке «Сеть» требуется только для контроля загрузки проекта по нужному адресу: при несовпадении с адресом, заданным в меню «Настройки» при компиляции выдается соответствующее предупреждение. Физически сетевой адрес в модуле CPU-03 задается DIP-переключателями.

5.2.3. Задание настроек модулей С-02.

После задания в контроллере серии ЭК-2000 модулей C-02 на вкладке «Сеть» для каждого заданного модуля появляется своя вкладка, войдя в которую, можно изменить режим модуля («Ведущий»/»Ведомый»). При выборе режима «Ведущий» появляется возможность выбора используемых каналов модуля C-02 (A, B или оба канала), а также параметров обмена данными через заданные каналы (скорость, тайм-аут и количество повторов). **Внимание! Двухканальный режим работы в режиме «Ведущий» в модулях C-02 с прошивками версии XC02MH20 и ниже не поддерживается!**

Кроме того, в режиме «Ведущий» имеется возможность задания карты заявок, обслуживающих устройства связи с объектом (УСО).

При нажатии правой кнопки мыши в поле «Заявки на чтение» или «Заявки на запись», на экране появляется контекстное меню, из которого выбирается необходимая операция с заявками.

Для задания новой заявки необходимо выбрать команду «Добавить…» из контекстного меню или нажать клавишу Ins в поле «Заявки на чтение» или «Заявки на запись». После этого на экране появляется окно «Новая заявка…». Для изменения параметров уже существующей заявки необходимо нажать клавишу Enter на строке заявки или дважды щелкнуть на ней мышью в поле «Заявки на чтение» или «Заявки на запись», или воспользоваться командой «Изменить…» контекстного меню. После этого на экране появляется окно «Изменить…».

В появившемся окне необходимо задать следующие параметры: адрес УСО, которое будет обсуживать заявка; регистр УСО, начиная с которого регистры будут считываться в модуль C-02 или записываться из модуля C-02; регистр модуля C-02, начиная с которого регистры будут считываться из УСО или записываться в УСО; количество считываемых или записываемых регистров; период сканирования – частота, с которой будет выполняться данная заявка (задается в количествах прохода карты заявок); время, в течение которого данная заявка не будет выпол-

няться, если УСО не готово (задается в количествах прохода карты заявок). Для сохранения заявки необходимо нажать кнопку «Сохранить», для отмены задания или изменения заявки – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по полям ввода параметров заявки, пока не установится на кнопке «Сохранить». При нажатии кнопки «Сохранить» выполниться проверка заданных параметров и, в случае положительного результата, добавление заявки к карте заявок, в противном случае – отображение сообщения об ошибке.

Для удаления заявки или выделенного блока заявок необходимо в поле «Заявки на чтение» или «Заявки на запись» нажать клавишу Del, или воспользоваться командой «Удалить» контекстного меню.

Для использования заявкой или блоком заявок другой области памяти модуля С-02 необходимо в поле «Заявки на чтение» или «Заявки на запись» нажать клавишу F2 или воспользоваться командой «Переместить…» контекстного меню. После этого на экране появляется окно «Начальный номер», в котором предлагается ввести новый регистр модуля С-02, начиная с которого регистры будут считываться из УСО или записываться в УСО. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости использования другой области памяти модуля С-02 и, в случае положительного результата, перемещение заявок, в противном случае – отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

Заявку или блок заявок можно помещать в буфер обмена, а затем вставлять их из буфера обмена в проект, что позволяет переносить карту заявок из одного проекта в другой, импортировать в проект карту заявок, заданную с помощью других приложений (например, Microsoft Excel), менять местами блоки заявок.

Для копирования заявки или блока заявок в буфер обмена необходимо нажать клавиши Ctrl-C или воспользоваться командой «Копировать» контекстного меню. Для перемещения заявки или блока заявок в буфер обмена необходимо нажать клавиши Ctrl-X или воспользоваться командой «Вырезать» контекстного меню.

Для вставки заявки или блока заявок в поле «Заявки на чтение» или «Заявки на запись» из буфера обмена необходимо нажать клавиши Ctrl-V или воспользоваться командой «Вставить» контекстного меню. При вставке на экране появляется окно «Начальный номер», как и при перемещении заявок. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости использования другой области памяти модуля С-02 и, в случае положительного результата, перемещение заявок, в противном случае – отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

При использовании для задания заявок другого приложения (например, Microsoft Excel) следует учитывать, что каждой заявке должны быть присвоены все параметры, кроме двух необязательных - периода сканирования и времени ожидания. Если последние два параметра не заданы, им присваиваются значения по умолчанию (1 и 100 соответственно).

В результате изменений карты заявок, если в нижней части окна присутствует и установлен флажок «Автоматически добавить процедуру конфигурации в программу», после нажатия на клавишу «ОК», сформируется файл с именем, совпадающим с именем проекта и расширением «.NET». В этом файле будет содержаться процедура, которая вызывает функции, настраивающие каналы модулей С-02 и задающая карту заявок. Данная процедура будет автоматически вызвана в начале работы про-

граммы. В пользовательской программе необходимо лишь запустить карту заявок с помощью функции FC02 (см. описание библиотеки SYSTEM.LIB).

5.3. Задание переменных и констант.

Для задания переменных и констант в ИСР ПО «CONT-Designer» используется окно данных, для вызова которого необходимо нажать клавиши Shift-F2, или кнопку «Окно данных» на панели управления, или воспользоваться командой меню «Окно | Окно данных». Появившееся окно включает три страницы: «Регистры и флаги», «Таймеры» и «Константы». При открытии окна, а также при нажатии правой кнопки мыши, на экране появляется контекстное меню, из которого выбирается необходимая операция с переменными или константами. Для перехода из окна редактора в окно данных и обратно служит клавиша F6.

Граница, отделяющая окно данных от окна редактора текста, является перемещаемой. Для перемещения границы влево/вправо необходимо подвести к ней курсор мыши и, после того как он изменит свою форму, перетащить границу в удобное положение.

Ряд команд контекстного меню применяются не только к одной переменной (константе), но и к блоку переменных или констант. Для выделения в окне данных блока следующих друг за другом элементов необходимо щелкнуть мышью на первом элементе блока, и, удерживая клавишу Shift, щелкнуть мышью на последнем элементе блока. Для выделения нескольких элементов блока, не следующих друг за другом, необходимо, удерживая клавишу Ctrl, щелкнуть мышью на каждом из элементов.

5.3.1. Задание регистров.

Для задания регистров в окне данных необходимо выбрать страницу «Регистры и флаги». Далее, для добавления нового регистра, необходимо выбрать команду «Добавить…» из контекстного меню или нажать клавишу Ins в окне данных. После этого на экране появляется окно «Новый регистр». Для изменения атрибутов уже существующего регистра необходимо нажать клавишу Enter на имени регистра или дважды щелкнуть на нем мышью в окне данных, или воспользоваться командой «Изменить…» контекстного меню. После этого на экране появляется окно «Изменить…».

В появившемся окне необходимо задать порядковый номер и имя регистра. Также, при необходимости, можно задать текстовый комментарий. При задании порядкового номера в поле ввода «Номер» можно записывать любое арифметическое выражение; при сохранении регистра оно автоматически вычисляется, и результат присваивается номеру регистра. Имеется возможность задания сразу нескольких регистров с последовательно возрастающими номерами, например РЕГ_10 с номером 10, РЕГ_11 с номером 11, РЕГ_12 с номером 12 и т.д. Для этого в поле ввода «Номер» необходимо ввести шаблон следующего вида:

[Начальный_номер–Конечный_номер],

например, [10-99], а в поле ввода «Имя» - шаблон вида:

Имя_регистра[Начальный_номер–Конечный_номер],

например, РЕГ_[10-99], при этом будут заданы 90 новых регистров. Для сохранения регистра необходимо нажать кнопку «Сохранить», для отмены задания нового или изменения существующего регистра – кноп-

ку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Для удаления регистра или блока регистров необходимо в окне данных нажать клавишу Del или воспользоваться командой «Удалить» контекстного меню.

Для перемещения регистра или блока регистров в другую область памяти, т.е. присвоения им других порядковых номеров, необходимо в окне данных нажать клавишу F2 или воспользоваться командой «Переместить…» контекстного меню. После этого на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться выделенный регистр или блок регистров. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения регистрам новых порядковых номеров и, в случае положительного результата, перемещение регистров, в противном случае - отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

Имеется возможность дублирования регистра или блока регистров, т.е. задания нового регистра или блока регистров с похожими именами. Для дублирования необходимо в окне данных нажать клавишу F5 или воспользоваться командой «Дублировать…» контекстного меню. После этого на экране появляется окно «Копия ИМЯ_РЕГИСТРА». В появившемся окне необходимо задать порядковый номер и имя для копии выделенного регистра или каждого регистра выделенного блока. Также, при необходимости, можно задать текстовый комментарий. Вместо номера может быть задано любое арифметическое выражение. Для сохранения копии регистра необходимо нажать кнопку «Сохранить», для отмены создания копии очередного регистра – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Регистр или блок регистров можно помещать в буфер обмена, а затем вставлять его из буфера обмена в проект, что позволяет переносить переменные из одного проекта в другой, импортировать в проект переменные, заданные с помощью других приложений, менять местами блоки регистров.

Для копирования регистра или блока регистров в буфер обмена необходимо нажать клавиши Ctrl-C или воспользоваться командой «Копировать» контекстного меню. Для перемещения регистра или блока регистров в буфер обмена необходимо нажать клавиши Ctrl-X или воспользоваться командой «Вырезать» контекстного меню.

Для вставки регистра или блока регистров в проект (окно данных) из буфера обмена необходимо нажать клавиши Ctrl-V или воспользоваться командой «Вставить» контекстного меню. При вставке на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться вставляемый регистр или блок регистров. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения регистрам новых порядковых номеров и, в случае положительного результата, вставка регистров, в противном случае - отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется. По

умолчанию в данном окне отображается текущий порядковый номер вставляемого регистра (или первого регистра во вставляемом блоке).

Следует отметить, что имеется возможность вставки блока регистров, заданных в приложении Microsoft Excel. В этом случае при копировании регистров в указанном приложении необходимо помещать в буфер обмена две или три колонки; первая колонка должна содержать порядковые номера регистров, вторая - их имена, третья (необязательная) - комментарии к регистрам.

Для поиска любой переменной (регистра, таймера, флага) по имени либо порядковому номеру или константы по имени либо значению в окне данных необходимо нажать клавиши Ctrl-F или воспользоваться командой «Найти…» контекстного меню. В появившемся окне «Найти» необходимо ввести требуемое имя, номер или значение, а также параметры поиска (поиск слова целиком или его части) и нажать клавишу Enter или кнопку «Найти далее». Если искомая переменная или константа будет найдена, осуществиться переход на соответствующую страницу окна данных и курсор установится на найденном элементе. Если искомый элемент не найден, на экран выводится соответствующее сообщение. Для продолжения поиска в окне данных необходимо нажать клавишу F3 или воспользоваться командой «Найти далее» контекстного меню. Для закрытия окна «Найти» необходимо нажать клавишу Esc или кнопку «Отмена».

Для замены имени регистра или определенного фрагмента в именах регистров в окне данных необходимо нажать клавиши Ctrl-R или воспользоваться командой «Заменить…» контекстного меню. В появившемся окне «Замена» необходимо ввести искомое имя (или фрагмент имени), а также новое имя (или фрагмент имени) и нажать клавишу Enter или кнопку «Найти далее». Если искомое имя (или фрагмент имени) будет найдено, курсор установится на найденном регистре. При нажатии кнопки «Заменить» имя (или фрагмент имени) найденного регистра заменится на новое. Для продолжения поиска и замены необходимо снова нажать кнопку «Найти далее» и, в случае нахождения регистра, нажать кнопку «Заменить». Такой способ удобен для выборочной замены фрагментов имен регистров. Для полной замены фрагментов имен регистров в окне «Замена» необходимо нажать кнопку «Заменить все». В этом случае во всех регистрах, содержащих в именах искомый фрагмент, выполнится замена этих фрагментов на новые без дополнительных подтверждений. Для закрытия окна «Заменить» необходимо нажать клавишу Esc или кнопку «Отмена».

5.3.2. Задание таймеров.

Для задания таймеров в окне данных необходимо выбрать страницу «Таймеры». Далее, для добавления нового таймера, необходимо выбрать команду «Добавить…» из контекстного меню или нажать клавишу Ins в окне данных. После этого на экране появляется окно «Новый таймер». Для изменения атрибутов уже существующего таймера необходимо нажать клавишу Enter на имени таймера или дважды щелкнуть на нем мышью в окне данных, или воспользоваться командой «Изменить…» контекстного меню. После этого на экране появляется окно «Изменить…».

В появившемся окне необходимо задать порядковый номер и имя таймера. Также, при необходимости, можно задать текстовый комментарий. При задании порядкового номера в поле ввода «Номер» можно записывать любое арифметическое выражение; при сохранении таймера

оно автоматически вычисляется, и результат присваивается номеру таймера. Имеется возможность задания сразу нескольких таймеров с последовательно возрастающими номерами, например ТАЙМЕР_10 с номером 10, ТАЙМЕР_11 с номером 11, ТАЙМЕР_12 с номером 12 и т.д. Для этого в поле ввода «Номер» необходимо ввести шаблон следующего вида: ***[Начальный_номер—Конечный_номер]***,
например, [10-99], а в поле ввода «Имя» - шаблон вида:
Имя_таймера[Начальный_номер—Конечный_номер],
например, ТАЙМЕР_[10-99], при этом будут заданы 90 новых таймеров. Для сохранения таймера необходимо нажать кнопку «Сохранить», для отмены задания нового или изменения существующего таймера – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Для удаления таймера или блока таймеров необходимо в окне данных нажать клавишу Del или воспользоваться командой «Удалить» контекстного меню.

Для перемещения таймера или блока таймеров в другую область памяти, т.е. присвоения им других порядковых номеров, необходимо в окне данных нажать клавишу F2 или воспользоваться командой «Переместить…» контекстного меню. После этого на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться выделенный таймер или блок таймеров. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения таймерам новых порядковых номеров и, в случае положительного результата, перемещение таймеров, в противном случае – отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

Имеется возможность дублирования таймера или блока таймеров, т.е. задания нового таймера или блока таймеров с похожими именами. Для дублирования необходимо в окне данных нажать клавишу F5 или воспользоваться командой «Дублировать…» контекстного меню. После этого на экране появляется окно «Копия ИМЯ_ТАЙМЕРА». В появившемся окне необходимо задать порядковый номер и имя для копии выделенного таймера или каждого таймера выделенного блока. Также, при необходимости, можно задать текстовый комментарий. Вместо номера может быть задано любое арифметическое выражение. Для сохранения копии таймера необходимо нажать кнопку «Сохранить», для отмены создания копии очередного таймера – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Таймер или блок таймеров можно помещать в буфер обмена, а затем вставлять его из буфера обмена в проект, что позволяет переносить переменные из одного проекта в другой, импортировать в проект переменные, заданные с помощью других приложений, менять местами блоки таймеров.

Для копирования таймера или блока таймеров в буфер обмена необходимо нажать клавиши Ctrl-C или воспользоваться командой «Копировать» контекстного меню. Для перемещения таймера или блока таймеров в буфер обмена необходимо нажать клавиши Ctrl-X или воспользоваться командой «Вырезать» контекстного меню.

Для вставки таймера или блока таймеров в проект (окно данных) из буфера обмена необходимо нажать клавиши Ctrl-V или воспользо-

ваться командой «Вставить» контекстного меню. При вставке на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться вставляемый таймер или блок таймеров. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения таймерам новых порядковых номеров и, в случае положительного результата, вставка таймеров, в противном случае - отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется. По умолчанию в данном окне отображается текущий порядковый номер вставляемого таймера (или первого таймера во вставляемом блоке).

Следует отметить, что имеется возможность вставки блока таймеров, заданных в приложении Microsoft Excel. В этом случае при копировании таймеров в указанном приложении необходимо помещать в буфер обмена две или три колонки; первая колонка должна содержать порядковые номера таймеров, вторая - их имена, третья (необязательная) - комментарии к таймерам.

Для поиска любой переменной (регистра, таймера, флага) по имени либо порядковому номеру или константы по имени либо значению в окне данных необходимо нажать клавиши Ctrl-F или воспользоваться командой «Найти…» контекстного меню. В появившемся окне «Найти» необходимо ввести требуемое имя, номер или значение, а также параметры поиска (поиск слова целиком или его части) и нажать клавишу Enter или кнопку «Найти далее». Если искомая переменная или константа будет найдена, осуществиться переход на соответствующую страницу окна данных и курсор установится на найденном элементе. Если искомый элемент не найден, на экран выводится соответствующее сообщение. Для продолжения поиска в окне данных необходимо нажать клавишу F3 или воспользоваться командой «Найти далее» контекстного меню. Для закрытия окна «Найти» необходимо нажать клавишу Esc или кнопку «Отмена».

Для замены имени таймера или определенного фрагмента в именах таймеров в окне данных необходимо нажать клавиши Ctrl-R или воспользоваться командой «Заменить…» контекстного меню. В появившемся окне «Замена» необходимо ввести искомое имя (или фрагмент имени), а также новое имя (или фрагмент имени) и нажать клавишу Enter или кнопку «Найти далее». Если искомое имя (или фрагмент имени) будет найдено, курсор установится на найденном таймере. При нажатии кнопки «Заменить» имя (или фрагмент имени) найденного таймера заменится на новое. Для продолжения поиска и замены необходимо снова нажать кнопку «Найти далее» и, в случае нахождения таймера, нажать кнопку «Заменить». Такой способ удобен для выборочной замены фрагментов имен таймеров. Для полной замены фрагментов имен таймеров в окне «Замена» необходимо нажать кнопку «Заменить все». В этом случае во всех таймерах, содержащих в именах искомый фрагмент, выполнится замена этих фрагментов на новые без дополнительных подтверждений. Для закрытия окна «Заменить» необходимо нажать клавишу Esc или кнопку «Отмена».

5.3.3. Задание флагов.

Для задания флагов в окне данных необходимо выбрать страницу «Регистры», установить курсор на необходимом регистре и воспользоваться командой «Окно флагов» из контекстного меню. После этого на

экране появляется окно «Флаги регистра ИМЯ_РЕГИСТРА». Для возврата на страницу регистров необходимо выбрать команду «Окно регистров» из контекстного меню.

Для изменения атрибутов уже существующего флага необходимо нажать клавишу Enter на имени флага или дважды щелкнуть на нем мышью в окне данных, или воспользоваться командой «Изменить…» контекстного меню. После этого на экране появляется окно «Изменить…».

В появившемся окне необходимо задать порядковый номер и имя флага. Также, при необходимости, можно задать текстовый комментарий. При задании порядкового номера в поле ввода «Номер» можно записывать любое арифметическое выражение; при сохранении флага оно автоматически вычисляется, и результат присваивается номеру флага. Имеется возможность задания сразу нескольких флагов с последовательно возрастающими номерами, например ФЛАГ_0 с номером 0, ФЛАГ_1 с номером 1, ФЛАГ_2 с номером 2 и т.д. Для этого в поле ввода «Номер» необходимо ввести шаблон следующего вида:

[Начальный_номер—Конечный_номер],

например, [0-15], а в поле ввода «Имя» - шаблон вида:

Имя_флага[Начальный_номер—Конечный_номер],

например, ФЛАГ_[0-15], при этом будут заданы 16 новых флагов. Для сохранения флага необходимо нажать кнопку «Сохранить», для отмены задания нового или изменения существующего флага – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Для удаления флага или блока флагов необходимо в окне данных нажать клавишу Del или воспользоваться командой «Удалить» контекстного меню.

Для перемещения флага или блока флагов в другую область памяти, т.е. присвоения им других порядковых номеров, необходимо в окне данных нажать клавишу F2 или воспользоваться командой «Переместить…» контекстного меню. После этого на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться выделенный флаг или блок флагов. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения флагам новых порядковых номеров и, в случае положительного результата, перемещение флагов, в противном случае – отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

Имеется возможность дублирования флага или блока флагов, т.е. задания нового флага или блока флагов с похожими именами. Для дублирования необходимо в окне данных нажать клавишу F5 или воспользоваться командой «Дублировать…» контекстного меню. После этого на экране появляется окно «Копия ИМЯ_ФЛАГА». В появившемся окне необходимо задать порядковый номер и имя для копии выделенного флага или каждого флага выделенного блока. Также, при необходимости, можно задать текстовый комментарий. Вместо номера может быть задано любое арифметическое выражение. Для сохранения копии флага необходимо нажать кнопку «Сохранить», для отмены создания копии очередного флага – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Номер» - поле ввода «Имя» - кнопка «Сохранить».

Для поиска любой переменной (регистра, таймера, флага) по имени либо порядковому номеру или константы по имени либо значению в окне данных необходимо нажать клавиши Ctrl-F или воспользоваться командой «Найти…» контекстного меню. В появившемся окне «Найти» необходимо ввести требуемое имя, номер или значение, а также параметры поиска (поиск слова целиком или его части) и нажать клавишу Enter или кнопку «Найти далее». Если искомая переменная или константа будет найдена, осуществиться переход на соответствующую страницу окна данных и курсор установится на найденном элементе. Если искомый элемент не найден, на экран выводится соответствующее сообщение. Для продолжения поиска в окне данных необходимо нажать клавишу F3 или воспользоваться командой «Найти далее» контекстного меню. Для закрытия окна «Найти» необходимо нажать клавишу Esc или кнопку «Отмена».

Для замены имени флага или определенного фрагмента в именах флагов в окне данных необходимо нажать клавиши Ctrl-R или воспользоваться командой «Заменить…» контекстного меню. В появившемся окне «Замена» необходимо ввести искомое имя (или фрагмент имени), а также новое имя (или фрагмент имени) и нажать клавишу Enter или кнопку «Найти далее». Если искомое имя (или фрагмент имени) будет найдено, курсор установится на найденном флаге. При нажатии кнопки «Заменить» имя (или фрагмент имени) найденного флага заменится на новое. Для продолжения поиска и замены необходимо снова нажать кнопку «Найти далее» и, в случае нахождения флага, нажать кнопку «Заменить». Такой способ удобен для выборочной замены фрагментов имен флагов. Для полной замены фрагментов имен флагов в окне «Замена» необходимо нажать кнопку «Заменить все». В этом случае во всех флагах, содержащих в именах искомый фрагмент, выполнится замена этих фрагментов на новые без дополнительных подтверждений. Для закрытия окна «Заменить» необходимо нажать клавишу Esc или кнопку «Отмена».

5.3.4. Задание констант.

Для задания константа в окне данных необходимо выбрать страницу «Константы». Далее, для добавления новой константы, необходимо выбрать команду «Добавить…» из контекстного меню или нажать клавишу Ins в окне данных. После этого на экране появляется окно «Новая константа». Для изменения атрибутов уже существующей константы необходимо нажать клавишу Enter на имени константы или дважды щелкнуть на ней мышью в окне данных, или воспользоваться командой «Изменить…» контекстного меню. После этого на экране появляется окно «Изменить…».

В появившемся окне необходимо задать значение и соответствующее имя константы. Также, при необходимости, можно задать текстовый комментарий. При задании значения можно записывать любое арифметическое выражение; при сохранении константы оно автоматически вычисляется, и результат присваивается значению константы. Имеется возможность задания сразу нескольких констант с последовательно возрастающими значениями, например КОНСТАНТА_10 с номером 10, КОНСТАНТА_11 с номером 11, КОНСТАНТА_12 с номером 12 и т.д. Для этого в поле ввода «Номер» необходимо ввести шаблон следующего вида:

[Начальное_значение—Конечное_значение],

например, [10-99], а в поле ввода «Имя» - шаблон вида:

Имя_константы[Начальное_значение—Конечное_значение],

например, КОНСТАНТА_[10-99], при этом будут заданы 90 новых кон-

стант. Для сохранения константы необходимо нажать кнопку «Сохранить», для отмены задания новой или изменения существующей константы – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Значение» - поле ввода «Имя» - кнопка «Сохранить».

Новые константы вставляются в окне данных перед константой, на которой установлен курсор. При добавлении новой константы автоматически рассчитывается ее порядковый номер в общем списке констант. Порядковые номера констант отображаются в окне данных справа от имен констант. Для просмотра порядковых номеров необходимо отодвинуть границу окна данных влево.

Для удаления константы или блока констант необходимо в окне данных нажать клавишу Del или воспользоваться командой «Удалить» контекстного меню.

Для перемещения константы или блока констант в другую область памяти, т.е. присвоения им других порядковых номеров, необходимо в окне данных нажать клавишу F2 или воспользоваться командой «Переместить…» контекстного меню. После этого на экране появляется окно «Начальный номер», в котором предлагается ввести новый порядковый номер, начиная с которого будет размещаться выделенная константа или блок констант. Вместо номера может быть задано любое арифметическое выражение. При нажатии в этом окне кнопки «ОК» или клавиши Enter выполниться проверка допустимости назначения константам новых порядковых номеров и, в случае положительного результата, перемещение констант, в противном случае – отображение сообщения об ошибке. При нажатии кнопки «Отменить» или клавиши Esc окно закрывается, и перемещение не выполняется.

Имеется возможность дублирования константы или блока констант, т.е. задания новой константы или блока констант с похожими значениями или именами. Для дублирования необходимо в окне данных нажать клавишу F5 или воспользоваться командой «Дублировать…» контекстного меню. После этого на экране появляется окно «Копия ИМЯ_КОНСТАНТЫ». В появившемся окне необходимо задать значение и имя для копии выделенной константы или каждой константы выделенного блока. Также, при необходимости, можно задать текстовый комментарий. Вместо номера может быть задано любое арифметическое выражение. Для сохранения копии константы необходимо нажать кнопку «Сохранить», для отмены создания копии очередной константы – кнопку «Отменить». При последовательном нажатии клавиши Enter курсор перемещается по следующим элементам: поле ввода «Значение» - поле ввода «Имя» - кнопка «Сохранить».

Константу или блок констант можно помещать в буфер обмена, а затем вставлять его из буфера обмена в проект, что позволяет переносить константы из одного проекта в другой, импортировать в проект константы, заданные с помощью других приложений, менять местами блоки констант.

Для копирования константы или блока констант в буфер обмена необходимо нажать клавиши Ctrl-C или воспользоваться командой «Копировать» контекстного меню. Для перемещения константы или блока констант в буфер обмена необходимо нажать клавиши Ctrl-X или воспользоваться командой «Вырезать» контекстного меню.

Для вставки константы или блока констант в проект (окно данных) из буфера обмена необходимо нажать клавиши Ctrl-V или воспользоваться командой «Вставить» контекстного меню. Следует отметить, что имеется возможность вставки блока констант, заданных в

приложении Microsoft Excel. В этом случае при копировании констант в указанном приложении необходимо помещать в буфер обмена две или три колонки; первая колонка должна содержать значения констант, вторая - их имена, третья (необязательная) - комментарии к константам.

Для поиска любой переменной (регистра, таймера, флага) по имени либо порядковому номеру или константы по имени либо значению в окне данных необходимо нажать клавиши Ctrl-F или воспользоваться командой «Найти…» контекстного меню. В появившемся окне «Найти» необходимо ввести требуемое имя, номер или значение, а также параметры поиска (поиск слова целиком или его части) и нажать клавишу Enter или кнопку «Найти далее». Если искомая переменная или константа будет найдена, осуществиться переход на соответствующую страницу окна данных и курсор установится на найденном элементе. Если искомый элемент не найден, на экран выводится соответствующее сообщение. Для продолжения поиска в окне данных необходимо нажать клавишу F3 или воспользоваться командой «Найти далее» контекстного меню. Для закрытия окна «Найти» необходимо нажать клавишу Esc или кнопку «Отмена».

Для замены имени константы или определенного фрагмента в именах констант в окне данных необходимо нажать клавиши Ctrl-R или воспользоваться командой «Заменить…» контекстного меню. В появившемся окне «Замена» необходимо ввести искомое имя (или фрагмент имени), а также новое имя (или фрагмент имени) и нажать клавишу Enter или кнопку «Найти далее». Если искомое имя (или фрагмент имени) будет найдено, курсор установится на найденной константе. При нажатии кнопки «Заменить» имя (или фрагмент имени) найденной константы заменится на новое. Для продолжения поиска и замены необходимо снова нажать кнопку «Найти далее» и, в случае нахождения константы, нажать кнопку «Заменить». Такой способ удобен для выборочной замены фрагментов имен констант. Для полной замены фрагментов имен констант в окне «Замена» необходимо нажать кнопку «Заменить все». В этом случае во всех константах, содержащих в именах искомый фрагмент, выполнится замена этих фрагментов на новые без дополнительных подтверждений. Для закрытия окна «Заменить» необходимо нажать клавишу Esc или кнопку «Отмена».

5.4. Написание текста программы.

Для редактирования текста уже существующей программы необходимо открыть проект и выбрать один из программных модулей (клавиши Ctrl-M, или кнопка «Выбрать модуль» на панели управления, или команда меню «Проект | Выбрать модуль…»).

Если программа еще не написана, необходимо создать новый проект. При этом автоматически генерируется главный программный модуль проекта, имеющий расширение .CON. Для создания новых программных модулей необходимо нажать клавиши Ctrl-N или воспользоваться командой меню «Файл | Создать». Затем следует сохранить созданный файл Noname.con под именем, совпадающим с именем проекта и указать ему необходимое расширение.

После открытия уже существующего или создания нового проекта интегрированная среда переходит в режим редактирования текста программы. При этом в правой части строки состояния отображается слово «Редактор». Ниже приводится список команд перемещения курсора:

На символ влево	←
На символ вправо	→
На слово влево	Ctrl ←
На слово вправо	Ctrl →
На строчку вверх	↑
На строчку вниз	↓
На страницу вверх	PgUp
На страницу вниз	PgDn
В начало строки	Home
В конец строки	End
В начало текста	Ctrl-PgUp
В конец текста	Ctrl-PgDn
Вправо по позициям табуляции	Tab

При редактировании текста программы можно многократно отменять последние выполненные действия (клавиши Ctrl-Z или команда меню «Правка | Отменить») или возвращаться к последним отмененным действиям (клавиши Ctrl-Y или команда меню «Правка | Вернуть»).

Для ускорения написания исходного текста программы и уменьшения количества ошибок в редакторе предусмотрена возможность работы с шаблонами. Шаблоны - это часто встречающиеся в программе фрагменты, которые можно вставить в исходный текст программы, начиная с текущей позиции курсора. Для этого необходимо вывести на экран список загруженных шаблонов (клавиши Ctrl-T или команда меню «Правка | Шаблоны…») и выбрать курсором один из них, либо нажать в окне редактора определенное сочетание клавиш, которое назначается каждому шаблону. Шаблоны хранятся в файле Default.txt, находящемся в директории ..\Program Files\Emicon\WinCont\Templates. В стандартном варианте они содержат названия команд языка CONT. Имеется возможность редактировать этот файл, изменяя существующие или добавляя новые шаблоны и назначая им соответствующие сочетания клавиш вызова.

Для ускорения написания программы применяется также перетаскивание имен переменных или констант из окна данных в текст программы. Для этого необходимо щелкнуть мышью на имени выбранной переменной или константы в окне данных, не отпуская кнопку, перетащить изменившийся курсор мыши в окно редактора и отпустить кнопку мыши. Имя выбранной переменной или константы вставится в исходный текст программы, начиная с текущей позиции курсора.

Любой фрагмент текста можно помещать в буфер обмена, а затем вставлять его из буфера обмена в текст программы, что позволяет менять расположения блоков текста, копировать повторяющиеся блоки текста, а также переносить их из одного программного модуля в другой и из одного проекта в другой.

Для копирования блока текста в буфер обмена необходимо нажать клавиши Ctrl-C или воспользоваться командой меню «Правка | Копировать». Для перемещения блока текста в буфер обмена необходимо нажать клавиши Ctrl-X или воспользоваться командой меню «Правка | Вырезать». Для вставки блока текста из буфера обмена в текст программы необходимо нажать клавиши Ctrl-V или воспользоваться командой меню «Правка | Вставить».

При нажатии в окне редактора правой кнопки мыши на экран выводится или окно, содержащее шаблоны, или стандартное контекстное меню, содержащее команды для работы с буфером обмена. Для выбора вида окна необходимо в меню «Настройки» установить или сбросить флаг «Стандартное контекстное меню».

Для поиска в программе необходимого фрагмента текста (например, имени операнда) в окне редактора необходимо нажать клавиши Ctrl-F или воспользоваться командой меню «Правка | Найти…». В появившемся окне «Найти» необходимо ввести искомый текст (по умолчанию в нем содержится слово, находящееся в позиции курсора), а также параметры поиска (поиск слова целиком или его части, с учетом или без учета регистра символов, направление вверх или вниз), и нажать клавишу Enter или кнопку «Найти далее». Если искомый текст будет найден, осуществится переход к началу найденного текста, который будет выделен цветом. Если искомый элемент не найден, на экран выводится соответствующее сообщение. Для продолжения поиска необходимо нажать в окне редактора клавишу F3, или в окне «Найти» - кнопку «Найти далее», или воспользоваться командой меню «Правка | Найти далее». Для закрытия окна «Найти» необходимо нажать клавишу Esc или кнопку «Отмена».

Для замены в программе одного фрагмента текста на другой необходимо нажать клавиши Ctrl-R или воспользоваться командой меню «Правка | Заменить…». В появившемся окне «Замена» необходимо ввести заменяемый и заменяющий фрагменты текста, а также параметры замены (замена слова целиком или его части, с учетом или без учета регистра символов), и нажать клавишу Enter или кнопку «Найти далее». Если заменяемый фрагмент текста будет найден, осуществится переход к началу найденного текста, который будет выделен цветом. При нажатии кнопки «Заменить» заменяемый фрагмент текста заменится на заменяющий. Для продолжения поиска и замены необходимо снова нажать кнопку «Найти далее» и, в случае нахождения фрагмента, нажать кнопку «Заменить». Такой способ удобен для выборочной замены фрагментов текста. Для полной замены фрагментов текста в окне «Замена» необходимо нажать кнопку «Заменить все». В этом случае во всем тексте, независимо от того, где установлен курсор, выполнится замена фрагментов без дополнительных подтверждений. Для закрытия окна «Заменить» необходимо нажать клавишу Esc или кнопку «Отмена».

При сохранении проекта помимо двоичного файла конфигурации с расширением .cpr генерируется также текстовый файл конфигурации с расширением .asc. Формат .asc-файла позволяет просматривать всю конфигурацию в текстовом редакторе, а также копировать в буфер обмена и переносить в открытый проект выделенные переменные, константы, модули DCS-2000 и заявки для модулей C-02. Изменение .asc-файла в текстовом редакторе не допускается. При отсутствии .cpr-файла допустимо открытие проекта путем выбора .asc-файла с помощью команды меню «Проект | Открыть…».

5.5. Компиляция и загрузка программы.

После создания или редактирования одного или нескольких программных модулей, необходимо запустить компилятор. Полная компиляция проекта применяется тогда, когда необходимо перекомпилировать все программные модули. Этот способ используется, если были изменения в функциях или драйверах, используемых в программе, или если необходимо сгенерировать листинговые файлы или ассемблерные эквиваленты модулей программы. Для полной компиляции программы необходимо нажать клавишу F9 или воспользоваться командой меню «Проект | Компилировать все», или кнопкой «Компилировать все» на панели управления.

Частичную компиляцию проекта можно использовать, если не бы-

ло изменений в функциях и драйверах, используемых в программе. В этом случае компилируются только измененные и более старшие по отношению к ним модули. Для частичной компиляции программы необходимо нажать клавиши Ctrl-F5 или воспользоваться командой меню «Проект | Компилировать».

После запуска компиляции автоматически выполняется сохранение программных модулей, заданных переменных, констант, а также аппаратной конфигурации контроллера. Процесс компиляции можно прервать нажатием клавиш Ctrl-Break.

Если в процессе компиляции обнаружены ошибки, в нижней части экрана отображается окно сообщений, в котором отображаются названия программных модулей, номера строк и соответствующие сообщения об ошибках. В таблице 7 приведены все возможные сообщения об ошибках; '*' - условное обозначение идентификатора, выводимого в тексте сообщения.

Таблица 7

Сообщение	Примечание
Неверное имя команды	Записано имя несуществующей команды
Слишком большая программа: необходимо разбиение на модули	Необходимо разбить Вашу программу (или модуль) на несколько модулей, так чтобы размер каждого исполняемого файла не превышал 32 Кбайт
Метка перезапуска уже определена	Программа может содержать не более одного оператора РЕСТАРТ, в котором определяется метка перезапуска
Типы операндов не совпадают	См. описание данной команды
Неверный тип операнда '*'	Операнд с именем '*' имеет недопустимый тип. См. описание данной команды
Метка '*' не найдена	Передается управление на несуществующую метку '*'
Запрещенный переход на метку '*'	• Команда перехода находится за пределами цикла, а метка с именем '*' - внутри цикла, либо наоборот; • Выход из процедуры (подпрограммы) на метку, находящуюся внутри цикла, либо в пределах другой процедуры (подпрограммы)
Неуместное использование метки	В данной точке программы запись метки недопустима
Переменная '*' доступна только для чтения	Попытка модификации переменной с именем '*', недоступной для записи
Метка '*' уже определена	Метка с именем '*' уже встречалась в программе

Значение вне допустимого диапазона	См. описание данной команды
Отсутствует '*'	Отсутствует ключевое слово, идентификатор или разделитель с именем '*'
Неверный литерал	
Нет идентификатора	
Неуместное использование команды	См. описание данной команды
Подпрограмма '*' обслуживает другой номер прерывания	Подпрограмма с именем '*' указана в нескольких командах ПРЕРЫВАНИЕ
Подпрограмма '*' не найдена	Подпрограмма с именем '*' указана в команде ПРЕРЫВАНИЕ, но отсутствует в текущем и в нижележащих модулях
Подпрограмма '*' не определена командой ПРЕРЫВАНИЕ	Если подпрограмма с именем '*' не используется в программе, закомментарьте ее, иначе - укажите в команде ПРЕРЫВАНИЕ
Подпрограмма '*' уже определена	Подпрограмма с именем '*' уже встречалась в программе
Процедура '*' уже определена	Процедура с именем '*' уже встречалась в программе
Процедура '*' не найдена	Процедура с именем '*' указана в команде ВЫЗВАТЬ, но отсутствует в текущем и в нижележащих модулях
Слишком много открытых библиотек	Количество подключаемых библиотек не должно превышать 50
Слишком много аргументов	Общее количество входных и выходных параметров функции или драйвера не должно превышать 500
Неверное количество операндов	См. описание данной команды
Операнд с именем '*' не определен	Если операнд с именем '*' является регистром, флагом, таймером или константой, определите его в окне данных; если портом или дискретным входом/выходом - задайте с помощью команды меню «Проект | Конфигурация»
Функция отсутствует в открытых библиотеках Драйвер отсутствует в открытых библиотеках	• Ошибка в написании имени или оформлении функции (драйвера); • Не подключена необходимая библиотека; • Функция (драйвер) не включена в

	подключенные библиотеки с помощью программы TCONTLIB.EXE
Компиляция прервана пользователем	Во время выполнения компиляции были нажаты клавиши Ctrl-Break
Фатальная ошибка компиляции	Необходимо посмотреть сообщения об ошибках, содержащиеся в файлах ASMSTATE.CMP и LNKSTATE.CMP, расположенных в каталоге ..\Program Files\Emicon\WinCont\Cmp
Слишком длинное имя идентификатора	Длина имени не должна превышать 25 символов
Неуместное использование }	Лишний символ '}'
Невозможно открыть файл <*>	Файл '*' невозможно считать с диска, либо его нельзя записать на диск (диск переполнен)
Нет памяти	Необходимо закрыть все выполняющиеся приложения, кроме Wincont

В открывшемся окне сообщений курсор устанавливается на первом сообщении об ошибке, при этом в окне редактора синий курсор указывает на строку, содержащую эту ошибку.

Для перехода к фрагменту текста, содержащему ту или иную ошибку, необходимо в окне сообщений дважды щелкнуть мышью на соответствующем сообщении. Также можно перейти в окно сообщений (для перехода из окна редактора в окно сообщений и обратно служит клавиша F6), перейти на строку, содержащую сообщение об ошибке и нажать клавишу Enter. После этого синий курсор установится в окне редактора на строке, содержащей эту ошибку.

Для перехода к следующему фрагменту текста, содержащему ошибку, необходимо, находясь в окне сообщений или окне редактирования, нажать клавиши Alt-F8. Для перехода к предыдущему фрагменту текста, содержащему ошибку, необходимо нажать клавиши Alt-F7.

После исправления всех ошибок необходимо снова откомпилировать программу. Если компиляция программы прошла без ошибок, генерируются исполняемые файлы с расширениями .Bxx (xx - номера модулей). Окно редактора после успешной компиляции переключается в режим просмотра исходного текста (для того, чтобы случайно не изменить текст программы в процессе отладки). В правой части строки состояния слово «Редактор» сменяется на «Просмотр», а в левой части отображается текст «Программа готова к загрузке в контроллер». Для возврата в режим редактирования необходимо нажать клавиши Alt-F10 или Ctrl-F10, или кнопку «Редактировать» на панели управления, или воспользоваться командой меню «Проект | Редактировать».

Перед началом загрузки программы в контроллер необходимо убедиться в правильности настроек обмена данными с контроллером и в нахождении последнего в режиме «Отладка» («Стоп»).

Внимание! Для перевода модулей CPU-11/15 с прошивкой версии XCC443 или выше, а также модулей CPU-17B в режим отладки и установки сетевых настроек каналов COM0 и COM1 в значения по умолчанию необходимо при включении питания модулей удерживать кнопку «J/D» в нажатом состоянии!

Для загрузки программы в контроллер необходимо нажать клави-

шу F9 или кнопку «Загрузить» на панели управления, или воспользоваться командой меню «Проект | Загрузить».

5.6. Отладка программы.

Если программа была скомпилирована с включенной отладочной информацией, то после загрузки исполняемых файлов в контроллер происходит инициализация выполнения программы, во время которой операционная система настраивается на выполнение первой команды программы. Курсор синего цвета в окне редактора устанавливается на строке, содержащей первую исполняемую команду программы. Теперь контроллер готов к выполнению программы и становятся доступными следующие возможности символьного отладчика:

- Непрерывное выполнение программы (клавиша F9 или кнопка «Выполнить» на панели управления, или команда меню «Отладка | Выполнить программу»). Для останова выполняемой программы необходимо нажать клавишу F12 или кнопку «Остановить» на панели управления, или воспользоваться командой меню «Отладка | Остановить программу».
- Выполнение команд в пошаговом режиме (клавиша F7 или кнопка «Выполнить шаг» на панели управления, или команда меню «Отладка | Выполнить шаг»). При этом за каждый шаг выполняются команды, расположенные на текущей строке.
- Выполнение программы до строки, помеченной курсором (для этого необходимо установить курсор на выбранной строке программы и нажать клавишу F4 или кнопку «Выполнить до курсора» на панели управления, или воспользоваться командой меню «Отладка | Выполнить до курсора»).
- Задание контрольных точек останова. Точкой останова считается первая команда, находящаяся на выбранной строке. Для задания точки останова необходимо выбрать необходимый программный модуль (клавиши Ctrl-M, или кнопка «Выбрать модуль» на панели управления, или команда меню «Проект | Выбрать модуль…»), установить курсор на выбранной строке программы и нажать Ctrl-F8 или воспользоваться командой меню «Отладка | Контрольная точка». При этом слева от выбранной строки отображается точка красного цвета. Таким образом можно пометить до 8-ми точек останова. Для снятия точки останова необходимо подвести курсор к выделенной строке и снова нажать Ctrl-F8 или воспользоваться командой меню «Отладка | Контрольная точка». Для снятия всех заданных точек останова необходимо нажать Shift-F8 или воспользоваться командой меню «Отладка | Сбросить все точки».
- Выполнение программы с остановом на контрольных точках (клавиша F5 или кнопка «Выполнять по точкам» на панели управления, или команда меню «Отладка | Выполнять по точкам»). При этом останов произойдет на ближайшей по ходу выполнения программы контрольной точке.
- Сброс программы (клавиши Ctrl-F2 или кнопка «Сбросить» на панели управления, или команда меню «Отладка | Сбросить программу»). При выполнении этой команды происходит инициализация выполнения программы, во время которой операционная система настраивается на выполнение первой команды программы. Курсор синего цвета в окне редактора устанавливается на строке, содержащей первую исполняемую команду программы.

Если программа была скомпилирована с выключенной отладочной информацией, то после загрузки исполняемых файлов в контроллер на экран выдается окно с запросом «Загрузка завершена. Запустить программу на выполнение?». При подтверждении запуска происходит инициализация выполнения программы, после которой программа сразу начинает выполняться в реальном масштабе времени без возможности ее останова. При отмене запуска окно редактора остается в режиме просмотра исходного текста с возможностью последующего запуска программы (клавиша F9 или кнопка «Выполнить» на панели управления, или команда меню «Отладка | Выполнить программу»).

Для перехода в режим редактирования программы необходимо нажать клавиши Alt-F10 или Ctrl-F10, или кнопку «Редактировать» на панели управления, или воспользоваться командой меню «Проект | Редактировать».

При выходе из интегрированной среды (клавиши Alt-F4 или команда меню «Файл | Выход») выполнение программы не прекращается.

Для постоянного контроля значений выбранных переменных используется окно слежения. Для вывода на экран окна слежения необходимо нажать клавиши Shift-F7 или кнопку «Окно слежения» на панели управления, или воспользоваться командой меню «Окно | Окно слежения». Для занесения переменных в окно слежения необходимо нажать клавиши Ctrl-F7, или воспользоваться командой меню «Отладка | Значение переменной…». После этого на экране появляется окно «Переменная» с полем ввода имени переменной. В качестве имени переменной можно задавать элемент массива регистров или таймеров. Если перед нажатием Ctrl-F7 установить курсор в окне редактора или в окне данных на имени интересующей переменной, в поле ввода будет отображаться это имя. После ввода или изменения имени переменной необходимо нажать клавишу Enter или кнопку «В окно слежения». В открывшемся окне «Вид представления» необходимо выбрать вид отображения: десятичный, шестнадцатеричный или двоичный. Если переменная является регистром, а ячейка памяти с более старшим адресом - регистром или элементом массива регистров, то можно также выбрать вещественное представление числа, записанного в эти два элемента. После нажатия клавиши Enter или кнопки «Применить» переменная заносится в окно слежения, при этом выводится ее имя и значение в выбранном представлении. При выполнении программы значения занесенных в окно слежения переменных модифицируются. Для удаления переменной из окна слежения необходимо установить курсор на удаляемой переменной и нажать клавишу Del.

Возможен также более быстрый способ занесения переменной в окно слежения. Для этого необходимо перетащить мышью имя выбранной переменной из окна данных в окно слежения, аналогично перетаскиванию в окно редактора при написании текста программы. После перетаскивания имени переменной сразу открывается окно «Вид представления».

Для изменения или единовременного просмотра значения переменной необходимо в окне «Переменная» нажать кнопку «Значение». После этого появляется окно «Значение» с текущим значением переменной, отображаемом в нескольких представлениях: десятичном, шестнадцатеричном, двоичном и, для регистров, вещественном. Для изменения значения переменной необходимо ввести новое число и нажать клавишу Enter. При задании значения в десятичном виде можно записывать любое арифметическое выражение; при нажатии на клавишу Enter оно автоматически вычисляется, и результат присваивается значению переменной.

Возможны также более быстрые способы просмотра или изменения текущего значения переменной: необходимо, находясь в окне редакто-

ра, в окне слежения или в окне данных, дважды щелкнуть мышью на имени переменной или нажать на нем клавишу Enter. После этого появляется окно «Значение».

Для закрытия окон «Значение», «Переменная» и «Вид представления» используется клавиша Esc.

5.7. Доступ к памяти данных контроллера.

Для просмотра и модификации области памяти данных контроллера используется ***окно «Память данных контроллера»***, для вызова которого необходимо нажать клавиши Shift-F9 или кнопку «Память данных» на панели управления, или воспользоваться командой меню «Окно | Память данных». В таблице одновременно отображаются по 120 ячеек памяти контроллера. Для перехода к отображению остальных ячеек используются клавиши управления курсором или кнопки в нижней части окна. Вид представления считываемых из контроллера 16-разрядных ячеек памяти (десятичное или шестнадцатеричное) задается выбором одной из кнопок в левой нижней части окна: «10», «16» или «2».

Для изменения значения ячейки памяти контроллера необходимо щелкнуть мышью на соответствующей ячейке таблицы, ввести новое значение и нажать клавишу «Enter». При задании значения в десятичном виде можно записывать любое арифметическое выражение; при нажатии на клавишу Enter оно автоматически вычисляется, и результат присваивается значению ячейки памяти. Для перехода к другой ячейке таблицы необходимо щелкнуть на ней мышью, или использовать клавиши управления курсором. Для быстрого перехода к просмотру (изменению) необходимой ячейки памяти необходимо в поле «Перейти» задать номер этой ячейки и нажать клавишу «Enter» (или дважды щелкнуть мышью в этом поле). При задании номера ячейки можно записывать любое арифметическое выражение.

Возможен также иной способ изменения значения ячейки памяти контроллера: для этого следует задать номер этой ячейки в поле «Адрес», ввести новое значение в поле «Значение» и нажать клавишу «Enter» (или дважды щелкнуть мышью в этом поле). При задании значения в десятичном виде и номера ячейки можно записывать любые арифметические выражения.

Следует отметить, что вызов окна памяти данных доступен на всех этапах написания и отладки программы, в том числе до открытия проекта и в процессе выполнения программы в контроллере. Это окно позволяет контролировать наличие и качество связи с контроллером. При наличии связи с контроллером в строке состояния окна памяти данных отображается слово «OK», а отсутствии или неудовлетворительной связи с контроллером – счетчик ошибок связи.

Для закрытия окна памяти данных необходимо нажать клавиши Alt-F4 или Shift-F9, или кнопку «Память данных» на панели управления, или воспользоваться командой меню «Окно | Память данных».

Для той же цели, что и окно памяти данных, служит ***утилита «Доступ к памяти данных контроллера»***, входящая в программную группу «CONT-Designer» («Пуск | Программы | CONT-Designer | Доступ к памяти данных»). Отличие этой утилиты от окна памяти данных в том, что она может работать независимо от системы программирования CONT-Designer. После вызова утилиты необходимо настроить параметры обмена данными с контроллером. Для выхода из нее используются клавиши Alt-F4.

5.8. Меню «Настройки».

После вызова меню «Настройки» на экране появляется окно «Настройки», разделенное на три группы: «Редактор», «Компиляция» и «Соединение». Значения настроек сохраняются при выходе из ИСР ПО CONT-Designer и автоматически загружаются при запуске последней.

В группе «Редактор» отображается текущий шрифт, используемый для отображения текста программы. Для изменения шрифта необходимо нажать кнопку «Изменить шрифт…». После этого открывается окно «Шрифт», в котором выбирается имя шрифта, его размер и цвет. После выбора наиболее удобных настроек шрифта необходимо нажать кнопку «OK». После этого во всех открытых программных модулях шрифт изменяется на новый. Рекомендуется выбирать моноширинный шрифт (например, Courier New). Кроме того, в группе «Редактор» задаются следующие настройки:

Стандартное контекстное меню - при нажатии в окне редактора правой кнопки мыши на экран выводится или окно, содержащее шаблоны (при сброшенном флажке), или стандартное контекстное меню, содержащее команды для работы с буфером обмена (при установленном флажке).

В группе «Компиляция» задаются следующие настройки:

Включить отладочную информацию – при установленном флажке напротив данной надписи при компиляции в исполняемые файлы программы записывается отладочная информация, поэтому отладка программы возможна лишь в этом режиме. После того, как программа полностью отлажена, перед загрузкой в контроллер ее рабочей версии, необходимо снять данный флажок для повышения быстродействия программы.

Генерировать ассемблерные файлы – при установленном флажке напротив данной надписи при компиляции на диске генерируются ассемблерные файлы программных модулей. Файлы имеют расширения .Axx (xx - номера модулей) и имена, совпадающие с именем программы.

Генерировать листинговые файлы – при установленном флажке напротив данной надписи при компиляции на диске генерируются листинговые файлы программных модулей. Файлы имеют расширения .Lxx (xx - номера модулей) и имена, совпадающие с именем программы.

Ошибки: остановиться после nn – число nn (от 1 до 99) определяет количество ошибок, после которого прекращается компиляция программы.

В группе «Соединение» задаются следующие настройки:

Адрес контроллера – здесь задается адрес контроллера, в который будут загружаться исполняемые файлы программы. Возможные значения адреса - от 1 до 255. При несовпадении адреса контроллера, указанного здесь и на вкладке «Сеть» меню «Проект | Конфигурация» перед компиляцией программы выдается предупреждающее сообщение. Если для связи выбран канал Ethernet, в этом поле задается IP-адрес контроллера или преобразователя интерфейсов CI-06B.

Канал связи – указывает канал, используемый для связи с контроллером (один из портов RS-232 или канал Ethernet, или один из каналов

RS-485 сетевой платы C-05/C-06).

Скорость, бод – здесь задается скорость обмена данными компьютера с контроллером при загрузке и отладке программы.

Тайм-аут – указывает число 10-миллисекундных интервалов, в течение которых должен приходить ответ компьютеру от контроллера при обмене данными между ними во время загрузки и отладки программы.

Повторов – здесь задается максимальное число повторов передачи сообщений от компьютера к контроллеру при обмене данными между ними во время загрузки и отладки программы. Если все попытки связи с контроллером закончились неудачно (нет ответа от контроллера, или обнаружена ошибка в ответе), формируется сообщение «Нет связи с контроллером».

Работать на фоне FIX – при установленном флажке напротив данной надписи возможна одновременная работа ИСР ПО CONT-Designer и SCADA-системы Fix фирмы Intelution. Если такая работа не предполагается, данный флажок должен быть сброшен (при запуске ИСР ПО CONT-Designer он сбрасывается).

Для сохранения заданных настроек ИСР ПО CONT-Designer следует нажать клавишу «OK» в нижней части окна, а для выхода без сохранения – клавишу «Отмена».

5.9. Информация о проекте.

В ИСР ПО CONT-Designer имеется возможность формирования и просмотра информации о проекте. Для вывода на экран окна с информацией о проекте следует воспользоваться командой меню «Проект | Информация…». Информацией о проекте является:

Номер версии проекта – состоит из двух чисел, разделенных точкой. Младшее число, находящееся справа от точки, увеличивается после компиляции измененного проекта, выполненной без включения отладочной информации. Старшее число, находящееся справа от точки, увеличивается при достижении младшим числом значения 1000. Старшее число можно также увеличить принудительно, нажав кнопку «Изменить». При каждом увеличении старшего числа младшее сбрасывается в нуль.

Контрольная сумма (CRC) проекта – значение в шестнадцатеричном представлении, модифицируется после компиляции измененного проекта.

Комментарий к проекту – представляет собой текстовый файл, содержащий любую текстовую информацию о проекте, например, перечень модификаций, сведения о разработчиках, о структуре проекта и т.д. Для просмотра и написания комментария к проекту следует в окне «Информация…» нажать кнопку «Комментарий…». После этого в окне редактора открывается текстовый файл с именем, совпадающим с именем проекта, и расширением «.TXT». После просмотра или редактирования комментария в окне редактора его можно закрыть (клавиши Ctrl-F4 или команда меню «Файл | Закрыть») или перейти в другое окно.

Для закрытия окна «Информация…» следует нажать кнопку «OK» или клавиши Alt-F4.

5.10. Выгрузка и загрузка регистров.

В ИСР ПО CONT-Designer имеется возможность выгрузки текущих значений регистров из контроллера и сохранения их на диске, а также загрузки сохраненных значений регистров в память контроллера (команды меню «Данные | Выгрузить» и «Данные | Загрузить»).

Данная возможность может быть полезной, например, при замене центрального процессорного модуля контроллера для сохранения уставок, состояний и текущих режимов работы объекта управления, хранящихся в памяти заменяемого модуля, и последующей их записи в память нового модуля.

5.11. Справочная система.

В ИСР ПО CONT-Designer встроена справочная система, включающая страницы содержания, предметного указателя и поиска. Справочная система содержит сведения по установки ИСР ПО CONT-Designer, информацию по работе с ЯП CONT, интегрированной средой, а также по написанию функций и драйверов. Для вызова справочной системы необходимо воспользоваться командой меню «? | Справка», или нажать клавишу F1. Далее в появившемся окне становятся доступными следующие страницы:

Содержание – здесь можно выбрать интересующий раздел, нажав на нем клавишу Enter или дважды щелкнув мышью, и выбрать необходимую страницу. Далее откроется окно с содержимым выбранной страницы. Для перехода к станицам содержания, предметного указателя, предыдущей страницы или печати текущей страницы служат соответствующие кнопки в верхней части окна.

Предметный указатель – здесь можно ввести или выбрать из списка интересующее ключевое слово, нажать клавишу Enter или кнопку «Показать» и просмотреть страницы, содержащие искомое ключевое слово.

Поиск - здесь можно ввести или выбрать интересующие слова из списка всех слов, содержащихся в файле справки. Далее необходимо выбрать нужную страницу, содержащую искомые слова, и нажать клавишу Enter или кнопку «Показать» для просмотра выбранной страницы.

Если установить курсор на том или ином операторе ЯП CONT, и вызвать справочную систему, то сразу откроется требуемая страница, содержащая информацию об этом операторе.

Для просмотра версии используемой ИСР ПО CONT-Designer необходимо воспользоваться командой меню «? | О программе…».