ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
СИСТЕМЫ СБОРА ТЕХНОЛОГИЧЕСКОЙ ИНФОРМАЦИИ

КОМПЛЕКС ПРОГРАММ ЗОНД

(Версия 4.40)

МОДУЛЬ “ВЫЧИСЛИТЕЛЬ”

Версия 4.40.0350

Руководство пользователя

Москва, 2016

СОДЕРЖАНИЕ

1. Общие сведения 7
1.1 Место вычислителя в комплексе ЗОНД 7
1.2 Использование вычислителя 7
1.3 Подготовка файлов исходных текстов 8
1.4 Трансляция исходных текстов 8
1.5 Выполняемый двоичный код 9
1.6 Ретрансляция двоичного кода 9
1.7 Выполнение двоичного кода 9
2. Препроцессор 10
2.1 Подключение внешних файлов: #INCLUDE 10
3. Данные 11
3.1 Переменные и константы 11
3.2 Типы данных 11
3.2.1 Константы 11
3.2.1.1 Вещественные константы 11
3.2.1.2 Двоичные константы 11
3.2.1.3 Шестнадцатеричные константы 11
3.2.2 Предопределённые константы 11
3.2.3 Внутренние переменные 12
3.2.4 Выходные переменные 12
3.2.5 Переменные из базы данных комплекса ЗОНД 13
3.3 Использование типов данных 14
3.4 Индексирование переменных 15
3.4.1 Индексы выходных переменных 15
3.4.2 Индексы переменных из базы данных 15
Таблица 3-1 16
Таблица 3-2 16
Таблица 3-3 17
4. Операции, выражения и операторы 18
4.1 Основные операции 18
4.1.1 Операция присваивания 18
4.1.2 Операция сложения 18
4.1.3 Операция вычитания 19
4.1.4 Операция изменения знака 19
4.1.5 Операция умножения 19
4.1.6 Операция деления 19
4.1.7 Операция возведения в степень 19
4.1.8 Поразрядные логические операции 20
4.1.8.1 Поразрядное И 20
4.1.8.2 Поразрядное ИЛИ 20
4.1.8.3 Поразрядное исключающее ИЛИ 20
4.1.9 Логическое НЕ 21

4.1.10 Порядок выполнения операций 21
4.2 Операторы 22
4.2.1 Оператор MESSAGE 22
4.2.2 Оператор MESREPER 23
4.2.3 Оператор INITOUTS 24
4.2.4 Оператор BEEP 24
4.2.5 Оператор SIREN_ON 24
4.2.6 Оператор SIREN_OFF 24
4.2.7 Оператор EXECUTE 24
4.2.8 Оператор SLEEP 24
4.2.9 Оператор SET 25
4.2.10 Оператор SET_STAT 26
4.2.11 Оператор SET_UST 27
4.2.12 Операторы LOCK_ARCH и UNLOCK_ARCH 27
4.3 Выражения 28

5. Выбор вариантов (ветвление) 29

5.1 Оператор IF 29
5.2 Расширение оператора IF с помощью ELSE 29
5.2.1 Выбор: конструкция IF-ELSE 29
5.2.2 Множественный выбор: конструкция ELSE-IF 30
5.2.3 Объединение операторов IF и ELSE 30
5.3 Оператор ENDIF 30

6. Функции 31

6.1 Функции пользователя 31
6.1.1 Создание функции: оператор FUNC 31
6.1.2 Аргументы функции 31
6.1.2.1 Определение функции с аргументом: формальные аргументы 31
6.1.2.2 Вызов функции с аргументом: фактические аргументы 32
6.1.2.3 Функция как “чёрный ящик” 32
6.1.2.4 Наличие нескольких аргументов 32
6.1.3 Возвращение значений функцией: оператор RETURN 33
6.1.4 Оператор ENDFUNC 33
6.2 Специальные функции пользователя ONINIT и ONSTOP 34
6.3 Встроенные функции 34
6.3.1 Математические функции 34
6.3.1.1 Функция ABS 34
6.3.1.2 Функция ACOS 34
6.3.1.3 Функция ASIN 34
6.3.1.4 Функция ATAN 34
6.3.1.5 Функция COS 35
6.3.1.6 Функция SIN 35
6.3.1.7 Функция TAN 35
6.3.1.8 Функция EXP 35
6.3.1.9 Функция LN 35
6.3.1.10 Функция LOG 35
6.3.1.11 Функция SQRT 35
6.3.1.12 Функция SIGN 35
6.3.1.13 Функция SGN 35
6.3.1.14 Функция POW 35
6.3.1.15 Функция RAND 35
6.3.1.16 Функция MIN 35

6.3.1.17 Функция MAX 35
6.3.1.18 Функция NMIN 36
6.3.1.19 Функция NMAX 36
6.3.1.20 Функция INT 36
6.3.1.21 Функция RESTDIV 36
6.3.2 Логические функции 36
6.3.2.1 Функция DOST 36
6.3.2.2 Функция TRUE 36
6.3.2.3 Функция FALSE 36
6.3.2.4 Функция GT 36
6.3.2.5 Функция GE 37
6.3.2.6 Функция LT 37
6.3.2.7 Функция LE 37
6.3.2.8 Функция EQ 37
6.3.2.9 Функция NE 37
6.3.2.10 Функция NOT 37
6.3.3 Функции над временем 38
6.3.3.1 Функция TIME 38
6.3.3.2 Функция SECOND 38
6.3.3.3 Функция MINUTE 38
6.3.3.4 Функция HOUR 38
6.3.3.5 Функция MONTHDAY 39
6.3.3.6 Функция MONTH 39
6.3.3.7 Функция YEAR 39
6.3.3.8 Функция WEEKDAY 39
6.3.3.9 Функция YEARDAY 39
6.3.3.10 Функция MAKETIME 39
6.3.4 Функции времени выполнения 39
6.3.4.1 Функция CYCLESEC 40
6.3.4.2 Функция EXECSEC 40
6.3.5 Функции над таймерами 40
6.3.5.1 Функция TIMERMSEC 40
6.3.5.2 Функция TIMERSEC 40
6.3.5.3 Функция TIMERMIN 40
6.3.5.4 Функция TIMERHOUR 40
6.3.5.5 Функция MAKETIMER 40
6.3.6 Функции счётчиков тиков 41
6.3.6.1 Функция GETTICKS 41
6.3.6.2 Функция TICKSIZE 41
6.3.7 Функция счётчик секунд CURRENTSEC 41
6.3.8 Функции перезагрузки 41
6.3.8.1 Функция STOP_SOFTDOG 41
6.3.8.2 Функция RESET 42
6.3.8.3 Функция REBOOT 42
6.3.9 Функции алгоритмов управления 42
6.3.9.1 Функция SET_WAIT 42
6.3.9.2 Функция SET_UNCOND 42
6.3.9.3 Функция PIDREG 43

7. Выполнение алгоритма 46

8. Панель инженера модуля “Вычислитель” 47

8.1.1 Дерево параметров конфигурации алгоблоков 47
8.1.2 Панель инструментов «Вычислитель» (Evaluator) 48
8.1.3 Горячие клавиши 49
8.1.4 Настройка УСО Вычислитель 50

8.1.5 Окно текстового редактора 51
8.1.6 Трансляция алгоблока 53
8.1.7 Пошаговая отладка алгоритма 54
8.1.8 Пример выполнения алгоритма по шагам 55

9. Перечень сообщений об ошибках 59

9.1 Сообщение транслятора 59
9.2 Сообщения вычислителя алгоблока 60
9.3 Сообщения ретранслятора 61
9.4 Сообщения при запуске алгоблока на выполнение 62

10. Список используемых документов 63

11. Приложения 64

11.1 Алфавитный указатель ключевых слов 64
11.1 Пример алгоритма 65

12. Краткий справочник функций 71

Как связаться с разработчиками?

Тел. \ факс.	**(495)382-56-34**
e-mail:	**zond@gpa.ru**
Web:	**http://www.gpa.ru/zond**
газовая связь:	**(700) 52-4-90,** (Москва, ул. Кирпичные выемки, д3.)

1. Общие сведения

1.1 Место вычислителя в комплексе ЗОНД

Комплекс программ ЗОНД позволяет отображать не только измеренные значения параметров, но и полученные расчётным путём. Параметры, значения которых вычислено по формулам, мы называем расчётными. Расчётные параметры могут быть аналоговыми, дискретными, датами или таймерами. Для написания формул расчёта и выполнения определённых действий в комплексе ЗОНД версии 4.40 и старше (выдачи сообщений, создания рапортов и т.п.) используется язык программирования “ФОР”. В качестве прототипа языка “ФОР” использован язык интерпретатора формул предыдущих версий комплекса ЗОНД.

Для выполнения расчётов и действий в комплексе ЗОНД предусмотрена возможность запуска до 16 задач-вычислителей, которые по своей организации соответствуют модулям работы с устройствами связи с объектом (УСО), такими как модули опроса систем телемеханики и различных контроллеров. Все эти задачи являются для базы данных комплекса ЗОНД импортёрами данных.

1.2 Использование вычислителя

УСО «Вычислитель» комплекса ЗОНД интерпретирует обработанные определённым образом исходные тексты на языке “ФОР” с задаваемой пользователем периодичностью.

Язык “ФОР” - язык “компилируемого” типа. Чтобы дать первое представление о процессе создания программы, ниже приводится упрощённая схема того, что необходимо сделать - начиная от написания программы и заканчивая её выполнением.

1 Используйте любой внешний редактор текстов или текстовый редактор в УСО «Вычислитель» для создания программы на языке “ФОР”.

2 Попытайтесь осуществить трансляцию вашей программы с помощью встроенного в УСО «Вычислитель» транслятора (кнопка «Транслировать» или **<F7>**). Он проведёт проверку правильности вашей программы и, если обнаружит ошибки, выдаст об этом сообщение. В противном случае транслятор выполнит перевод программы в некоторый внутренний язык вычислителя (двоичный код) и поместит результат в файл конфигурации соответствующего алгоблока вычислителя.

3 Вы можете запустить программу на выполнение вручную клавишей **<F5>**. Оттранслированная и помещённая в файл конфигурации вычислителя программа будет запускаться автоматически при запусках комплекса ЗОНД, если с помощью в конфигурации ЗОНД включён автозапуск соответствующего алгоблока.

Далее рассмотрим шаги процесса создания программы более подробно.

1.3 Подготовка файлов исходных текстов

В данном разделе рассматривается процедура создания исходного текста программ с помощью встроенного редактора.

Для запуска редактора необходимо запустить УСО «Вычислитель» и выбрать нужный алгоблок. Затем открыть существующий файл (кнопка «Открыть» с диска»), ретранслировать файл из алгоблока (кнопка «Ретранслировать») или начать создавать новый файл в окне текстового редактора.

Редактор формул представляет собой обычный текстовый редактор, с помощью которого пишутся и редактируются тексты программ расчёта значений параметров. Для удобства написания формул в редактор встроены сервисные функции.

Первая из них, окно **«Вставка служебных слов»** - активизируется нажатием правой кнопки мыши в текстовом редакторе. Меню позволяет не набирая, а просто выбирая из пунктов меню сносить в текст программы функции, операции и операторы.

Меню содержит следующие пункты:

* функции;
* операции;
* операторы;
* дополнительно;
* константы;
* препроцессор.

Функции	- полный перечень встроенных функций, поддерживаемых языком “ФОР”.
Операции	- перечень выполняемых арифметических и логических операций.
Операторы	- условные операторы, операторы вывода сообщений, включения/выключения звуковой сигнализации, выполнения файлов-заданий, а также задержки.
Дополнительно	- перечень допустимых индексов и их значения.
Константы	– перечень предопределённых ключевых слов, заменяющих константы.
Препроцессор	– перечень директив препроцессора.

Вторая функция - окно **«Вставка реперов»** - активизируется вводом “{“ в текстовом редакторе. Окно позволяет не набирая, а просто выбирая из списка (или вводя номер репера) вносить в текст программы реперы, содержащиеся в базе данных комплекса ЗОНД.

1.4 Трансляция исходных текстов

Транслятор преобразует текст программы в бинарные данные - команды, которые могут быть выполнены вычислителем. В процессе трансляции эти данные выводятся в специальное окно (для визуального контроля разработчиком). В случае успешной трансляции у пользователя запрашивается необходимость сохранения результатов в конфигурационном файле (вопрос можно отключить в настройках).

1.5 Выполняемый двоичный код

Полученный в результате трансляции двоичный код при записи в конфигурационный файл снабжается контрольной суммой, которая анализируется при последующих запусках задач-вычислителей, а также при просмотре конфигурации алгоблока.

1.6 Ретрансляция двоичного кода

Текст программы может быть восстановлен из двоичного кода путём его ретрансляции. Для этого в комплекс ЗОНД встроен декомпилятор. Для вызова ретранслятора необходимо нажать клавишу **<F8>** или нажать на панели управления кнопку “Ретранслировать”.

1.7 Выполнение двоичного кода

Задачи-вычислители на основе кода, созданного компилятором, производят вычисления значений переменных. Результаты расчётов заносятся либо во временные переменные, либо в выходные переменные вычислителя, в этом случае они становятся доступными для задачи обработки и занесения в базу данных.

Кроме того, в процессе отладки текста программы, откомпилированный и не содержащий синтаксических ошибок код может быть выполнен в пошаговом режиме с отображением промежуточных результатов.

Далее рассмотрим возможные типы данных в языке “ФОР”.

2. Препроцессор

Трансляция файла исходного кода выполняется в два этапа. Вначале работает препроцессор, затем транслятор.

2.1 Подключение внешних файлов: #INCLUDE

При помощи директивы #include «FILE_1.evl» можно помещать содержимое внешнего файла в транслируемый файл. Текст из внешнего файла включается в транслируемый файл вместо директивы #include на этапе компиляции.

Путь к файлу не обходимо указать относительно корневой директории проекта.

Пример:

Расположение папок внутри проекта программы «Конфигуратор БД» (см. ***Док. 4***):

PROJECT\BASE_0
PROJECT\EVL\lib_1.evl
PROJECT\ZONDVIZA
PROJECT\project.zdb
PROJECT\drv.znd

Для включения внешнего файла lib_1.evl необходимо написать в коде:

#Include «EVL\lib_1.evl»

Внешний файл, в свою очередь, сам может включать в себя другой файл. Путь к файлу также должен быть указан относительно корневой директории проекта (***Док. 4***).

3. Данные

3.1 Переменные и константы

Вычислитель, выполняя программу, может заниматься разнообразной деятельностью. Он может складывать числа, сортировать их, выводить сообщения, управлять звуковой сигнализацией, делать логические выводы. Чтобы заниматься всем этим, программам необходимо работать с “данными” - числами и датами, т.е. объектами, которые несут в себе информацию, предназначенную для использования. Некоторые данные устанавливаются равными определённым значениям ещё до того, как программа начнёт выполнятся, а после её запуска сохраняют такие значения неизменными на всем протяжении работы программы. Это “константы”. Другие данные могут изменяться, или же им могут быть присвоены значения во время выполнения программы; они называются “переменными”.

3.2 Типы данных

3.2.1 Константы

Компилятор «Вычислителя» обладает следующим свойством. Если при записи константы внутрь ошибочно будет записан недопустимый символ, например латинская буква, константа будет считаться переменной.

3.2.1.1 Вещественные константы

Вещественные константы в вычислителе представлены в виде чисел с плавающей точкой. В качестве разделителя целой и дробной части выступает точка (‘.’), причём в числах без дробной части её присутствие не обязательно.

Например: 137 80.21

3.2.1.2 Двоичные константы

В исходном коде могут быть использованы двоичные константы. Двоичная константа состоит из цифр ‘0’ и ‘1’. В конце указан суффикс ‘B’ что значит binary (двоичная константа).

Например: 1010b 0110B

3.2.1.3 Шестнадцатеричные константы

Шестнадцатеричная константа записывается шестнадцатеричными цифрами ‘0’…‘F’, в конце указывается суффикс ‘H’ (hexadecimal constant).

Например: FFFFh 3B8H

3.2.2 Предопределённые константы

Предопределённые константы – константы заданные ключевыми словами. Использование ключевых слов делает код понятнее. Список предопределённых констант приводится ниже:

NVG	- нижняя возможная граница;
NAG	- нижняя аварийная граница;
NTG	- нижняя технологическая граница;
VTG	- верхняя технологическая граница;
VAG	- верхняя аварийная граница;
VVG	- верхняя возможная граница;
STATUS_TREATMENT	- статус “Обработка разрешена”;
STATUS_CAPTURE	- статус “Датчик исправен”;
STATUS_MESWIN	- статус “Сообщения в окно”;
STATUS_PRN	- статус “Сообщения на печать”;
STATUS_PRFILE	- статус “Сообщения в файл”;
CURRENT_KVIT	- параметр квитирован;
STATUS_KVIT_GOOD	- статус “Квитирование улучшения”;
STATUS_KVIT_BAD	- статус “Квитирование ухудшений”;
STATUS_SIREN	- статус “Включить сирену”;
STATUS_ALARM	- статус “Аварийный”;
STATUS_EQUIP	- статус “Аппарат в работе”;
ALL_IN_BASE	- все параметры в Базе Данных.

Предопределённые константы удобно использовать в операторах и вызовах функций.

Например: SET_UST {Рвх К19}[SYS_NUM],VTG,45.4

3.2.3 Внутренние переменные

Внутренние переменные - переменные, предназначенные для промежуточного хранения результатов расчёта при обработке выражений одного алгоблока. Переменные локальны для каждого алгоблока. Значения внутренних переменных после окончания расчёта сохраняются до следующего цикла выполнения, то есть они статические.

Максимальная длина имени переменной равна 15 символам. Имя переменной может включать в себя русские и латинские буквы, цифры, а также символ подчёркивания (‘_’). Строчные и прописные буквы транслятором не различаются, то есть переменная ‘ab’ и ‘AB’ будут интерпретироваться как одна переменная. Для вычислений в одном алгоблоке максимально может быть заведено 32 внутренних переменных.

3.2.4 Выходные переменные

Выходные переменные - переменные, предназначенные для долговременного хранения результатов расчётов внутри алгоблока. Значения выходных переменных после окончания расчёта сохраняются до следующего цикла, а также при перезапусках задач-вычислителей и всего комплекса ЗОНД в целом.

Выходные переменные предназначены также для передачи результатов расчётов в базу данных комплекса ЗОНД, при этом значение переменной, которая связана с аналоговым параметром, не должно превышать допустимых величин,

указанных в шкале паспорта параметра. Если это произошло, то значение параметра не изменится, но приобретёт признак недостоверности. Чтобы избежать таких ситуаций, позаботьтесь о корректном задании шкал расчётных параметров в базе данных ЗОНД.

Дискретные расчётные параметры могут принимать значения строго в соответствии с их битовым размером - 1, 2 (для дискретных) или 3 (для восьмипозиционных) бита.

Выходные переменные имеют фиксированные имена, которые определяют их тип:

AOUTXXXXX - аналоговые переменные;
DOUTXXXXX - дискретные переменные,

где XXXXX - номер переменной 00001 - 65535, причём незначащие нули могут быть опущены.

Переменные с именами AOUTXXXXX могут хранить числа с плавающей запятой с точностью 14-15 десятичных цифр в диапазоне плюс/минус (10^{-307}-10^{308}), что соответствует типу double в языке C (IEEE 754-2008).

Переменные с именами DOUTXXXXX могут хранить целые числа в диапазоне -128 - +127, что соответствует типу char в языке C (8 бит).

Переменные с именами AOUTXXXXX могут также хранить даты, полученные с использованием функций работы со временем, при этом в них хранятся целые числа в диапазоне 0 - (2^{32}-1), что соответствует типу unsigned long в языке C.

Массивы выходных переменных могут быть проинициализированы при помощи оператора INITOUTS (см. 4.2.3).

3.2.5 Переменные из базы данных комплекса ЗОНД

Значение параметра из базы данных комплекса ЗОНД в выражениях обозначается через его репер (краткое наименование). Значения параметров и признаки достоверности для вычисления формулы берутся из базы данных. При заполнении (создании) базы данных необходимо помнить о том, что для каждого параметра репер должен быть строго индивидуален, то есть два различных параметра не могут иметь одинаковые реперы.

Репер параметра может содержать русские и латинские буквы, цифры, пробелы (‘ ‘), символ подчёркивания (‘_’), а также символы ‘,’, ‘:’, ‘#’. Следует отметить, что при использовании символа ‘#’ в репере параметра, он не должен быть первым символом. С другой стороны, в реперы не должны включаться знаки арифметических и логических операций (‘+’, ‘-‘, ‘*’, ‘/’, ‘!’, ‘&’, ‘|’, ‘~’), знак присвоения (‘=’), символ ‘%’, а также круглые и квадратные скобки (‘(‘, ‘)’, ‘[‘, ‘]’). Круглые парные скобки ‘()’ влияют на порядок вычисления выражений, а следовательно, и выражения в целом.

В случаях, когда в реперах переменных из базы данных все же присутствуют «запрещённые» символы, в том числе символы-разделители и знаки операций

(пробел, ‘+’, ‘-‘, ‘*’, ‘!’ и т.д.), репер необходимо заключать в фигурные скобки, например:

{AB-021/17}

Если такой репер сносится в текст программы через меню подсказок, он будет заключён в фигурные скобки автоматически. Тоже самое произойдёт при ретрансляции объектного кода.

Переменную из базы данных в тексте программы можно также идентифицировать следующей записью:

#целая_константа, где

#	- символ номера,
целая_константа	- целая константа, определяющая системный номер параметра в базе данных

Или

РЕПЕР[*выражение*], где

РЕПЕР	- репер параметра,
выражение	- выражение в квадратных скобках, задающее смещение системного номера, например:

C=TBX12[75+i*2]

В этом примере переменной C присвоится значение переменной из базы данных комплекса ЗОНД с системным номером, определяемым как сумма системного номера переменной TBX12, константы 75 и произведения переменной i на константу 2.

3.3 Использование типов данных

Во время разработки программы необходимо составить для себя список требуемых внутренних переменных, которых не может быть более 32. Имена переменных выбирайте таким образом, чтобы они указывали на их смысл. Длина имени может достигать 15 символов. Перед использованием внутренних переменных в правой части выражений не забывайте их проинициализировать, так как их значения не сохраняются после остановки вычислителя или при выходе из комплекса программ ЗОНД.

Следует отметить, что хотя выходные переменные AOUT и DOUT имеют разные имена, при совпадении их номеров адресуют одну и ту же область памяти, т.е. записав в переменную DOUT003 значение 25, получим значение переменной AOUT003, равное 25.0 и наоборот. В случае если значение, записанное в переменную AOUT, превышает допустимый диапазон для типа DOUT, значение, прочитанное из переменной DOUT для проведения каких-либо операций будет случайным.

3.4 Индексирование переменных

3.4.1 Индексы выходных переменных

Поскольку выходные переменные в вычислителе являются массивом, обращение к ним может производится по индексам. Индексами выходных переменных могут являться:

* константы в диапазоне 1-65535;
* другие переменные;
* выражения,

например:

```
AOUT[13]
DOUT[23+N]
AOUT[DOUT[X+Y]+7-L]
```

Индексированные выходные переменные могут появляться как в правой, так и в левой частях выражений или выступать аргументами функций.

3.4.2 Индексы переменных из базы данных

Под параметрами с индексами понимаются элементы выражения следующего вида:

РЕПЕР[*индекс*], где

РЕПЕР	- репер параметра,
индекс	- константа, задаваемая ключевым словом в квадратных скобках

Значения допустимых индексов и их смысл для разных типов параметров приведены в Таблица 3-1 и Таблица 3-2.

Индексированные переменные из базы данных могут появляться только в правой части выражения или выступать аргументами функций.

Таблица 3-1

Тип параметра	Индекс			
	0	1	2	3
Аналоговый	текущее значение	с начала суток всего	за прошлые сутки	с начала месяца
Дискретный	текущее значение	нет	нет	нет
Измеритель-ная линия	с начала суток, в т.ч. недостоверно	с начала суток, в т.ч. достоверно	за прошлые сутки	с начала месяца
Дата-Время	текущее значение	нет	нет	нет

Перечень ключевых слов переведён в Таблица 3-2.

Таблица 3-2

Инд.	Ключевое слово	Обозначает
0	FLOW_TODAY_FALSE	Интегральное значение с начала суток по недостоверным данным
1	FLOW_TODAY_TRUE	Интегральное значение с начала суток по достоверным данным
2	FLOW_PREVDAY	Интегральное значение за прошлые сутки
3	FLOW_MONTH	Интегральное значение с начала месяца
4	SYS_NUM	Системный номер параметра
5	FLOW_TODAY_TOTAL	Интегральное значение с начала суток всего
6	NVG	Нижняя возможная граница (уставка)
7	NAG	Нижняя аварийная граница (уставка)
8	NTG	Нижняя технологическая граница (уставка)
9	DELTA	Дельта (величина коррекции технологических границ)
10	VTG	Верхняя технологическая граница (уставка)
11	VAG	Верхняя аварийная граница (уставка)
12	VVG	Верхняя возможная граница (уставка)
13	NNG	Номер нарушенной границы
14	SCALEB	Начало шкалы параметра
15	SCALE	Длина шкалы параметра
16	STATUS_KVIT_GOOD	Статус - признак необходимости квитировать улучшение значений параметров;
17	STATUS_MESWIN	Статус - признак необходимости вывода сообщений в окно протокола событий;
18	STATUS_PRN	Статус - признак необходимости вывода сообщений на печать;
19	STATUS_PRFILE	Статус - признак необходимости вывода сообщений в файл протокола событий;
20	STATUS_KVIT_BAD	Статус - признак необходимости квитировать ухудшение значений технологических

Инд.	Ключевое слово	Обозначает
		параметров;
21	STATUS_TREATMENT	Статус - признак запрета/разрешения обработки опрошенных значений;
22	STATUS_CAPTURE	Статус - признак исправности датчика;
23	CURRENT_KVIT	Признак квитирования параметра. 1 - параметр квитирован, 0 – нет;
24	STATUS_SIREN	Статус “Включить сирену”;
25	STATUS_ALARM	Статус “Аварийный”;
26	STATUS_EQUIP	Статус “Аппарат в работе”.

Значения номеров нарушенных границ в зависимости от типа параметра в базе данных приведены в Таблица 3-3.

Таблица 3-3

Тип параметра	Значение	Обозначает
	3	нарушена верхняя возможная граница
	2	нарушена верхняя аварийная граница
	1	нарушена верхняя технологическая граница
Аналоговый	0	параметр в норме
	-1	нарушена нижняя технологическая граница
	-2	нарушена нижняя аварийная граница
	-3	нарушена нижняя возможная граница
Счетчик времени, счетчик импульсов, внешний таймер	0	параметр в норме
	1	нарушена граница контроля

Примеры использования индексированных переменных приведены ниже:

N=PEPER[SYS_NUM] - получить системный номер параметра с репером «PEPER», и сохранить его в переменной N;

ST=PEPER[STATUS_KVIT_BAD] - получить значение признака необходимости квитирования ухудшения значения параметра с репером «PEPER», и сохранить его в переменной ST;

G= PEPER[NNG] - получить номер нарушенной границы параметра с репером «PEPER», и сохранить его в переменной G.

4. Операции, выражения и операторы

4.1 Основные операции

Операции в языке “ФОР” применяются для представления арифметических и логических действий.

4.1.1 Операция присваивания

В языке “ФОР” знак равенства не означает “равно”. Он означает операцию присваивание присваивания некоторого значения. С помощью оператора

C = 2002

переменной с именем **C** присваивается значение 2002, т.е. элемент слева от знака = - это *имя* переменной, а элемент справа - её *значение*. Мы называем символ = “операцией присваивания”. Ещё раз хотим обратить ваше внимание на то, что смысл указанной строки не выражается словами “**C** равно 2002”. Вместо этого нужно говорить так: “присвоить переменной **C** значение 2002”. В этой операции действие выполняется справа налево.

Оператор вида

2002 = C

на языке “ФОР” не имеет смысла, поскольку **2002** - число. Вы не можете присвоить константе какое-то значение; её значением является она сама. Поэтому помните, что элемент, стоящий слева от знака =, всегда должен быть именем переменной (может быть с индексом). Для употребления правильных названий понятий скажем, что вместо использованного ранее термина “элемент” обычно употребляют слово “операнд”. Операнды - это то, над чем выполняются операции.

В одной строке программы на языке “ФОР” не может быть более одной операции присваивания.

4.1.2 Операция сложения

‘+’	- сложение	c = a+b

Выполнение операции + приводит к сложению двух величин, стоящих слева и справа от этого знака. Операнды могут быть как переменными, так и константами.

Операция + называется “бинарной”, что отражает тот факт, что она имеет дело с *двумя* операндами.

4.1.3 Операция вычитания

‘-‘ - вычитание c = a-b

Выполнение операции вычитания приводит к вычитанию числа, расположенного справа от знака -, из числа, стоящего слева от этого знака.

4.1.4 Операция изменения знака

‘-‘ - унарный минус c = -b

Знак минус используется также для указания или изменения алгебраического знака некоторой величины. Когда знак минус используется подобным образом, данная операция называется “унарной”. Такое название указывает на то, что она имеет дело только с одним операндом.

4.1.5 Операция умножения

‘*’ - умножение c = a*b

Операция умножения обозначается знаком *. При выполнении этой операции значение переменной **a** умножается на значение переменной **b**, и результат присваивается переменной **c**.

4.1.6 Операция деления

‘/’ - деление c = a/b

В языке “ФОР” символ / указывает на операцию деления. Величина, стоящая слева от этого знака, делится на величину, расположенную справа от него.

Заметим, что операция деления всегда проводится как с данными с плавающей точкой, т.е. результат будет числом с плавающей точкой.

4.1.7 Операция возведения в степень

‘^’ - возведение в степень c = a^b

Операция возведения в степень обозначается знаком ^. При выполнении этой операции значение переменной **a** возводится в степень, в качестве которой используется значение переменной **b**, и результат присваивается переменной **c**.

4.1.8 Поразрядные логические операции

Следующие три операции производятся над данными, полученными из целой части значений переменных или выражений (напомним, что величины в вычислителе хранятся в формате с плавающей точкой). Они называются “поразрядными”, потому что выполняются отдельно над каждым разрядом независимо от разряда, находящегося слева или справа.

4.1.8.1 Поразрядное И

‘&’ - И (AND) c = a & b

Эта бинарная операция сравнивает последовательно разряд за разрядом два операнда. Для каждого разряда результат равен 1, если только оба соответствующих разряда операндов равны 1 (в терминах “истинно - ложно” результат получается истинным, если только каждый из двух одноразрядных операндов является истинным). Так,

(10010011) & (00111101) == (00010001)

потому что только четвёртый и первый разряды обоих операндов содержат 1.

4.1.8.2 Поразрядное ИЛИ

‘|’ - ИЛИ (OR) c = c | 01H

Эта бинарная операция сравнивает последовательно разряд за разрядом два операнда. Для каждого разряда результат равен 1, если любой из соответствующих разрядов операндов равен 1 (в терминах “истинно - ложно” результат получается истинным, если один из двух или оба одноразрядных операндов является истинным). Так,

(10010011) | (00111101) == (10111111)

потому что все разряды, кроме шестого, в одном из двух операндов имеют значение 1.

4.1.8.3 Поразрядное исключающее ИЛИ

‘~’ - исключающее ИЛИ (XOR) c = a ~ b

Эта бинарная операция сравнивает последовательно разряд за разрядом два операнда. Для каждого разряда результат равен 1, если один из двух (но не оба) соответствующих разрядов операндов равен 1 (в терминах “истинно -ложно”

результат получается истинным, если один из двух (но не оба) одноразрядных операндов является истинным). Поэтому

(10010011) ~ (00111101) == (10101110)

Заметим, что, поскольку нулевой разряд в обоих операндах имеет значение 1, нулевой разряд результата имеет значение 0.

4.1.9 Логическое НЕ

‘!’ - НЕ (NOT) c = !b

Операция ! имеет один операнд, расположенный справа. Результат логического НЕ имеет значение “истина” если операнд имеет значение “ложь”, и наоборот.

4.1.10 Порядок выполнения операций

Рассмотрим следующую строку:

c = 25 + 60*n/SCALE

В этом операторе имеются операции сложения, умножения и деления. Какая операция будет выполнена первой?

Совершенно очевидно, что изменение порядка выполнения действий может приводить к различным результатам, поэтому язык “ФОР” нуждается в наборе непротиворечивых правил, указывающих, какое действие осуществлять первым. Язык “ФОР” делает это, задавая приоритет той или иной операции. Каждой операции назначается уровень старшинства. Умножение и деление имеют более высокий уровень, чем сложение и вычитание, поэтому они выполняются первыми. Если же две операции имеют один и тот же уровень старшинства, они выполняются в том порядке, в котором присутствуют в строке. Для большинства операций обычный порядок - слева направо, (операция = является исключением из этого правила).

Если вы захотите, чтобы сложение выполнялось перед делением, тогда надо написать строку по-другому:

c = (25 + 60*n)/SCALE

В первую очередь выполняется все, что заключено в скобки; внутри скобок действуют обычные правила.

Приведём таблицу операций, располагая их по приоритетам и показывая порядок выполнения.

Таблица 4-1

Операции (от высшего приоритета к низшему)	Порядок выполнения
( )	Л -> П
! -(унарный)	П -> Л
* /	Л -> П
+ -	Л -> П
&	Л -> П
~	Л -> П
|	Л -> П
=	П -> Л

Условные обозначения: П -> Л - порядок выполнения справа налево, Л -> П - наоборот.

4.2 Операторы

Под операторами в языке “ФОР” понимаются команды, выдаваемые комплексу ЗОНД на выполнение определённых действий. Выполнение операторов непосредственно не влияет на ход самих вычислений (за исключением операторов выбора вариантов, которые будут рассмотрены в главе 5).

4.2.1 Оператор MESSAGE

Оператор вывода сообщений в протокол событий имеет следующий синтаксис:

MESSAGE format[,arg1[,arg2[,arg3[,arg4[,arg5[,arg6]]]]]],

где:

format - текстовая строка, ограниченная кавычками («), может содержать процент (%), признак начала формата вывода числа:

%[flags][width][.precision]format type

Формат подробно описан в Таблица 4-2.

Таблица 4-2

Название поля	Возможные значения
flags	-(минус) - сместить влево, без отступа; + (плюс) - всегда помещать знак числа;
width	число знакомест, отводимое для печати числа, включая знак и десятичную точку;
precision	число знаков после десятичной точки (запятой);
format type	f - число с фиксированной десятичной точкой; e,E - научная нотация записи числа; g,G - компактная форма (какой формат короче, e или g);

Например:

MESSAGE «#%4.0f РАСХОД (Q=%6.2f)»,РАСХОД[4],РАСХОД

В протокол будет выведено приблизительно следующее сообщение:

СЕН 09 1998 18:11:42 #0024 РАСХОД (Q=1234.12)

Сообщение всегда сопровождается текущими датой и временем. Максимальное количество выводимых чисел - 6. То есть, знак % в строке формата может встречаться не более шести раз.

Данные параметров будут приведены к формату с плавающей точкой, поэтому в строке формата нужно употреблять модификации %f.

4.2.2 Оператор MESREPER

Оператор вывода в протокол сообщений, связанных с параметром, имеет следующий синтаксис:

MESSAGE sys, format[,arg1[,arg2[,arg3[,arg4[,arg5[,arg6]]]]]],

где:

sys	Системный номер параметра
format	Аналогично оператору MESSAGE

Например:

MESREPER {ТС8 КП64}[SYS_NUM],"ЗНАЧЕНИЕ НЕДОСТОВЕРНО"

В протокол будет выведено следующее сообщение:

СЕН 09 2007 18:11:42 #00024 ТС8 КП64 ЗНАЧЕНИЕ НЕДОСТОВЕРНО

Сообщение всегда сопровождается текущими датой и временем. Максимальное количество выводимых чисел - 6. То есть, знак % в строке формата может встречаться не более шести раз. Сообщение начинается с системного номера

и репера параметра в том же формате, в каком следуют сообщения задачи обработки об изменениях значений параметров.

4.2.3 Оператор INITOUTS

Оператор INITOUTS позволяет проинициализировать массив выходных переменных. Синтаксис:

INITOUTS first_index,value,count

где:

first_index	индекс первой инициализируемой выходной переменной (1…);
value	значениие которым будут проинициализированы выходные переменные;
count	число проинициализированных выходных переменных.

4.2.4 Оператор BEEP

BEEP()	- выдать звуковой сигнал (однократный короткий “бип”).

4.2.5 Оператор SIREN_ON

SIREN_ON()	- включить звуковой сигнал “сирена”.

4.2.6 Оператор SIREN_OFF

SIREN_OFF()	- выключить звуковой сигнал “сирена”.

4.2.7 Оператор EXECUTE

Оператор выполнения командного файла имеет следующий синтаксис:

EXECUTE [*путь*]*имя_файла*, где:

путь	- путь к командному файлу;
имя_файла	- имя командного файла

Формат выполняемых командных файлов комплекса ЗОНД и перечень возможных команд описан в “Описании применения” комплекса ЗОНД.

4.2.8 Оператор SLEEP

SLEEP(*выражение*)	- задержка выполнения программы на время, заданное выражением (в единицах по 55 мс).

Не рекомендуем использовать задержки более чем на 2,9 секунды, так как в этом случае при завершении работы комплекса ЗОНД могут не сохранится значения выходных переменных алгоблока, выполняющего задержку.

4.2.9 Оператор SET

В текстах программ для вычислителя можно применять оператор SET:

```
SET системный_номер, значение
```

например:

```
SET ДАВЛЕНИЕ[SYS_NUM],30
SET ДАВЛЕНИЕ[SYS_NUM],ДАВЛЕНИЕ1
SET НАСОС[SYS_NUM],0
SET НАСОС[SYS_NUM],1
SET НАСОС[SYS_NUM],НАСОС1          ; перевести НАСОС
                                   ; в такое же состояние
                                   ; как и НАСОС1
SET КРАН[SYS_NUM],0
SET КРАН[SYS_NUM],1
```

при этом интерпретация поля «значение» следующая (Таблица 4-3):

Таблица 4-3

Тип параметра	Значение	Смысл действий
Дискретный однобитный	0	Команда «Отключить»
	1	Команда «Включить»
Дискретный двухбитный	0	Команда «открыть»
	1	Команда «закрыть»

В Таблица 4-4 рассматриваются варианты работы оператора

```
SET НАСОС[SYS_NUM],0
```

в зависимости от поля «Тип» статуса параметра.

Таблица 4-4

Тип в статусе параметра	Результат работы оператора SET
Дискретный	никакого
Ручной ввод	никакого
Управляемый извне	значение засылается в УСО в соответствии с подключением
Устанавливаемый извне	значение заносится непосредственно в базу данных комплекса Зонд

Два последних типа в Таблица 4-4 можно установить только при разрешении на управление и занесение значений извне в файле zondviza.cfg.

Оператор выполняется асинхронно. Успешное завершение в случае засылки в УСО не гарантируется.

4.2.10 Оператор SET_STAT

В текстах программ для установки значений статусов параметров в базе данных (см. ***Док. 2***) можно применять оператор SET_STAT:

SET_STAT системный_номер, статус (предопределённая константа), значение (0 или 1)

Например:

```
SET_STAT ОХРАНА_ББ[SYS_NUM],STATUS_KVIT_GOOD,0
SET_STAT {ОБЪЕКТ 1}[SYS_NUM],STATUS_TREATMENT,1
```

Напомним, что статусы параметров влияют на работу задачи обработки.

Оператору SET_STAT обязательно указывается три параметра. Первый, системный номер параметра, которому устанавливается значение статуса. Если операцию нужно произвести над всеми параметрами в Базе данных, можно использовать предопределённую константу ALL_IN_BASE.

Например:

```
SET_STAT ALL_IN_BASE,STATUS_KVIT_GOOD,1
```

Вторым параметром, модифицируемый статус (см. Таблица 4-5).

Таблица 4-5

Индекс статуса	Обозначает
STATUS_KVIT_GOOD	необходимость квитирования улучшения значения параметра («1» - квитировать необходимо, «0» - нет);
STATUS_MESWIN	необходимость вывода сообщений в окно протокола событий («1» - выводить, «0» - нет);
STATUS_PRN	необходимость вывода сообщений на печать («1» - выводить, «0» - нет);
STATUS_PRFILE	необходимость вывода сообщений в файл протокола событий («1» - выводить, «0» - нет);
STATUS_KVIT_BAD	необходимость квитирования ухудшения значения параметра («1» - квитировать, «0» - нет);
STATUS_TREATMENT	запрет («0») /разрешение («1») обработки опрошенных значений;
STATUS_CAPTURE	исправность датчика («1» - исправен, «0» - нет);
STATUS_SIREN	Необходимость включать звуковой сигнал («1» - включать, «0» - нет);
STATUS_ALARM	Аварийный сигнал («1» - аварийный, «0» - нет);
STATUS_EQUIP	Аппарат (сигнал) в работе – «1», выведен из работы «0»;

Индекс статуса	Обозначает
CURRENT_KVIT	Текущее состояние квитирования («1» - квитирован, «0» - нет); установка флага из алгоблока обновляет текущее состояние звука согласно общему правилу обработки Зонд “есть неквитированные параметры – есть звук”

Третий параметр, устанавливаемое значение статуса параметра. Допустимые значения - «0» и «1».

4.2.11 Оператор SET_UST

В текстах программ для установки значений уставок параметров в базе данных (см. ***Док. 2***) можно применять оператор SET_UST:

SET_UST системный_номер, тип уставки (предопределённая константа) (см Таблица 4-6), значение уставки

Например:

```
SET_UST {P BX}[SYS_NUM],NVG,0
SET_UST ДАВЛЕНИЕ[SYS_NUM],VVG,45
```

Оператору SET_UST обязательно указывается три параметра. Первый, системный номер параметра, которому задаётся уставка. Вторым параметром, тип уставки (см. Таблица 4-6). Третьим – значение уставки. Значения уставок должно укладываться в шкалу параметра.

Таблица 4-6

Предопределенная константа	**Значение**
NVG	Нижняя возможная граница (уставка)
NAG	Нижняя аварийная граница (уставка)
NTG	Нижняя технологическая граница (уставка)
VTG	Верхняя технологическая граница (уставка)
VAG	Верхняя аварийная граница (уставка)
VVG	Верхняя возможная граница (уставка)

4.2.12 Операторы LOCK_ARCH и UNLOCK_ARCH

Операторы обеспечивают синхронизацию доступа к каналу связи с задачами УСО. В пределах текста алгоблока между вызовами lock_arch и unlock_arch задача УСО, которая осуществляет много командную операцию взятия порции архивных данных устройства, не делает эту операцию. В это время подобную операцию может сделать код УСО Вычислитель. Единственная реализация механизма – с УСО modbus мастер, тип архивов PGC90.50.

LOCK_ARCH	- обозначить начало зоны	lock_arch uso,dir,
UNLOCK_ARCH	работы с архивом	unlock_arch uso,dir

- обозначить конец зоны работы с архивом	uso – номер интерфейса УСО (с 0), dir – номер линии УСО (с 1)

4.3 Выражения

Комплекс программ ЗОНД позволяет вычислять значения аналоговых, дискретных и параметров типа “ДАТА-ВРЕМЯ” на основе известных значений параметров (возможно даже других расчётных). Последовательность действий для получения значения расчётного параметра определяется формулой расчёта, которая, в общем случае, может быть записана в несколько строк. Каждая строка должна содержать символ присваивания <’=’> (за исключением строк операторов). Число строк не ограничено. Максимальная длина строки 408 символов, а размер оттранслированной формулы (бинарного кода) - не более 32кб.

На протяжении первых глав мы использовали термин “выражение”. Как уже отмечалось, программа на языке “ФОР” состоит из строк; большинство же строк состоят из выражений. Исходя из этого, вначале разумно рассмотреть выражения, что мы и сделаем.

Общий вид строки программы, не содержащей операторов, приведён ниже:

Внутренняя переменная = Выражение[реперы, переменные, константы, функции]
Внешняя переменная = Выражение[реперы, переменные, константы, функции]

Выражение представляет собой объединение операций и операндов. Простейшее выражение состоит из одного операнда; отталкиваясь от него, вы можете строить более сложные конструкции.

Нетрудно заметить, что операнды могут быть константами, переменными или их сочетаниями. Некоторые выражения состоят из меньших выражений, которые мы можем назвать подвыражениями.

Важным свойством языка “ФОР” является то, что каждое выражение имеет значение. Чтобы определить это значение, мы выполняем операции в порядке, определяемом уровнями старшинства.

Текст программы может содержать комментарии. Символ начала комментария ‘;’ (точка с запятой). Комментарий распространяется от точки с запятой до конца строки. Комментарии в двоичном коде не хранятся, поэтому ретранслированный из двоичного кода текст программы их содержать не будет (рекомендуем хранить исходные тексты программ с комментариями).

5. Выбор вариантов (ветвление)

Согласно теории вычислительных систем существуют три следующие формы управления процессом выполнения программ:

1 Выполнение последовательности строк.
2 Выполнение определённой последовательности строк до тех пор, пока некоторое условие истинно.
3 Использование проверки истинности условия для выбора между различными возможными способами действия.

Язык “ФОР” обеспечивает реализацию первой и третьей форм.

Первая форма управления не представляет трудностей, поэтому рассмотрим только третью форму.

5.1 Оператор IF

```
IF      <выражение>     - начало условного блока
        выражение1      if NE(a,b)   если выражение истинно (a!=b),
        ....            a = b        то выполняется a = b
        выражениеN      endif
ENDIF                   - конец условного блока
```

При рассмотрении условных операторов обычно под выражением понимают условное выражение; с его помощью сравниваются значения двух величин. Если такое выражение истинно, то условный блок строк выполняется. В противном случае он пропускается. Вообще говоря, в качестве условия может быть использовано любое выражение, и если его значение равно 0, то оно считается ложным, если нет - истинным.

5.2 Расширение оператора IF с помощью ELSE

5.2.1 Выбор: конструкция IF-ELSE

```
IF      <выражение>     - начало условного блока
        выражение1      if LT(a,b)   если выражение истинно (a<b),
        ...             c = a        то выполняется c = a,
        выражениеN      else         в противном случае c = b
ELSE                    c = b
        выражение1      endif
        ...
        выражениеM
ENDIF                   - конец условного блока
```

5.2.2 Множественный выбор: конструкция ELSE-IF

```
IF          <выражениеА>     - начало условного блока
            выражение1       if LT(a,b)      если выражение А истинно (a<b),
            ...                c = a         то выполняется c = a,
            выражениеN       else            в противном случае
ELSE                           if GT(a,b) если выражение Б истинно (a>b),
  IF        <выражениеБ>         c = b       то выполняется c = b,
            выражение1         else            в противном случае
            ...                  c = 0         выполняется c = 0
            выражениеK         endif
  ELSE                       endif
            выражение1
            ...
            выражениеM
  ENDIF
ENDIF                        - конец условного блока
```

5.2.3 Объединение операторов IF и ELSE

Для разрешения вопроса о том, какому оператору **if** соответствует какой оператор **else**, существует правило, которое гласит, что **else** соответствует ближайшему **if** (транслятор не обращает внимания на отступу в строках).

5.3 Оператор ENDIF

Каждому оператору **if** должен соответствовать оператор **endif**, определяющий конец условно выполняемого блока строк, даже если этот блок состоит из одной строки.

6. Функции

Функция - самостоятельная единица программы, спроектированная для реализации конкретной задачи. Функции в языке “ФОР” играют ту же роль, какую играют функции, подпрограммы и процедуры в других языках, хотя детали их структуры могут быть разными. Вызов функции приводит к выполнению некоторых действий или получению некоторых величин, используемых затем в программе.

Функции избавляют от повторного программирования, если конкретную задачу необходимо выполнить в программе несколько раз. Даже в том случае, если некоторая задача выполняется в программе только один раз, лучше оформить ее решение в виде функции, поскольку функции повышают уровень модульности программы и, следовательно, облегчают ее чтение, внесение изменений и коррекцию ошибок (естественно, при этом приходится немного пожертвовать скоростью выполнения программы).

В языке “ФОР” существует набор встроенных функций, назначение и поведение которых жёстко фиксировано, а также существует возможность их создания самим пользователем. Число функций пользователя, написанных в одной программе, не должно превышать 32.

6.1 Функции пользователя

Функции пользователя разрабатываются для выполнения требуемых в конкретном приложении действий и вычислений и могут выполняться только в том алгоблоке, в тексте программы которого они определены.

6.1.1 Создание функции: оператор FUNC

Для определения функции пользователя в программе используется оператор FUNC:

```
FUNC имя_функции ()
```

где *имя_функции* - уникальное имя функции в программе, по которому затем могут производится обращения к ней.

Правила формирования имени функции такие же, как и для имен внутренних переменных (см. П. 3.2.3).

Описание функции может появляться в тексте программы в любом месте, но не позже обращения к ней и не внутри описания другой функции.

6.1.2 Аргументы функции

Для большинства случаев необходима возможность передачи функции некоторой величины, которая будет влиять либо на ход ее выполнения, либо на результаты вычислений. Такая величина называется аргументом функции.

6.1.2.1 Определение функции с аргументом: формальные аргументы

Пусть определение нашей функции начинается со строки

```
FUNC space (number)
```

При этом переменная **number** называется “формальным” аргументом. Фактически это новая переменная, для которой в алгоблоке вычислителя выделена отдельная ячейка.

Посмотрим, как можно пользоваться этой функцией.

6.1.2.2 Вызов функции с аргументом: фактические аргументы

Задача в этом случае состоит в том, чтобы присвоить некоторую величину формальному аргументу **number**. После того, как эта переменная получит свое значение, программа сможет выполнить свою задачу. Переменной **number** при вызове функции присваивается значение фактического аргумента.

Рассмотрим использование функции **space()**.

```
Blank = space(25)
```

Фактический аргумент здесь 25, и эта величина присваивается формальному аргументу - переменной **number**, т.е. вызов функции оказывает следующее действие:

```
number = 25
```

Короче говоря, формальный аргумент - переменная в вызываемой программе, а фактический аргумент - конкретное значение, присвоенное этой переменной вызывающей программой. Фактический аргумент может быть константой, переменной или даже более сложным выражением. Независимо от этого фактический аргумент сначала вычисляется, а затем его величина (в данном случае число) передаётся функции.

6.1.2.3 Функция как “чёрный ящик”

Функцию можно рассматривать как “чёрный ящик”; её задают через поступающую информацию (вход), и полученные результаты (выход). Все, что происходит внутри чёрного ящика, можно не рассматривать до тех пор, пока не нужно будет писать саму программу, реализующую эту функцию. Вход связан с функцией через аргумент, выход - через возвращаемое значение, о котором речь пойдёт дальше.

6.1.2.4 Наличие нескольких аргументов

Если для связи с некоторой функцией требуется более одного аргумента, то наряду с именем функции можно задавать список аргументов, разделённых запятыми, как показано ниже:

```
FUNC difference (a,b)
return (a-b)
ENDFUNC
```

Максимальное число формальных аргументов функции - 32.

6.1.3 Возвращение значений функцией: оператор RETURN

Входные величины в функциях могут обрабатываться благодаря наличию аргументов; выходная же величина возвращается (выдаётся) при помощи ключевого слова RETURN:

```
RETURN выражение
```

Возвращаемое значение можно присвоить переменной или использовать как часть некоторого выражения, например:

```
value = user_func(z)
answer = 2*user_func(z) + 25
message «%8.0f»,user_func(z)
```

Оператор **return** оказывает и другое действие. Он завершает выполнение функции и передаёт управление в вызывающую программу. Это происходит даже в том случае, если оператор **return** является не последним оператором тела функции.

Вы можете также использовать просто оператор

```
RETURN
```

Его применение приводит к тому, что функция, в которой он содержится, завершает своё выполнение и управление возвращается в вызывающую функцию. Поскольку у данного оператора отсутствует выражение, возвращаемое функцией значение не определено и может быть равным совершенно произвольному числу. Тоже самое произойдёт и при отсутствии в функции оператора **return.**

Следует отметить, что в языке “ФОР”, даже если функция пользователя не должна возвращать конкретного значения, строка вида

```
user_func(z)
```

вызовет сообщение об ошибке при трансляции, т.е. в такой строке обязательно должен присутствовать оператор присваивания

```
value = user_func(z)
```

6.1.4 Оператор ENDFUNC

Для указания транслятору окончания описания функции применяется оператор ENDFUNC. Оператор состоит только из одного слова, его присутствие в функции в отличие от оператора RETURN обязательно.

6.2 Специальные функции пользователя ONINIT и ONSTOP

Во время работы код программы выполняется циклически. Часто возникает потребность выполнить часть программы один раз при инициализации и перед завершением. Для этого в языке “ФОР” предусмотрены специальные пользовательские функции ONINIT и ONSTOP. Функция ONINIT выполняется перед началом работы цикла алгоритма, ONSTOP – при завершении работы.

```
FUNC ONINIT (T)
  ; здесь код функции
ENDFUNC

FUNC ONSTOP (T)
  ; здесь код функции
ENDFUNC

  ; далее основной код алгоритма
```

Обе функции имеют один аргумент, время вызова функции (в формате функции TIME).

6.3 Встроенные функции

Как уже отмечалось, в языке “ФОР” существует набор встроенных функций, которые могут использоваться в программах, выполняющихся в разных алгоблоках. Эти функции делятся на четыре группы:

* математические функции;
* логические функции;
* функции над временем;
* функции над таймерами

Рассмотрим эти группы более подробно.

6.3.1 Математические функции

6.3.1.1 Функция ABS

ABS - взятие модуля c = abs(a)

6.3.1.2 Функция ACOS

ACOS - арккосинус c = acos(a)

6.3.1.3 Функция ASIN

ASIN - арксинус c = asin(a)

6.3.1.4 Функция ATAN

ATAN - арктангенс c = atan(a)

6.3.1.5 Функция COS

COS	- косинус	c = cos(a)

6.3.1.6 Функция SIN

SIN	- синус	c = sin(a)

6.3.1.7 Функция TAN

TAN	- тангенс	c = tan(a)

6.3.1.8 Функция EXP

EXP	- экспонента	c = exp(a)

6.3.1.9 Функция LN

LN	- натуральный логарифм	c = ln(a)

6.3.1.10 Функция LOG

LOG	- десятичный логарифм	c = log(a)

6.3.1.11 Функция SQRT

SQRT	- квадратный корень	c = sqrt(a)

6.3.1.12 Функция SIGN

SIGN	- взятие знака	c = sign(a), c = 1, если a < 0 и c = 0, если a > 0

6.3.1.13 Функция SGN

SGN	- взятие знака	c = sgn(a), c = -1, если a < 0; c = 0, если a = 0 и c = 1, если a > 0

6.3.1.14 Функция POW

POW	- возведение в степень	c = pow(a,b)

6.3.1.15 Функция RAND

RAND	- получить псевдослучайное число	c = rand()

6.3.1.16 Функция MIN

MIN	- минимум из двух чисел	c = min(a,b), c = a, если a <= b и c = b, если a > b

6.3.1.17 Функция MAX

MAX	- максимум из двух чисел	c = max(a,b), c = a, если a >= b и c = b, если a < b

6.3.1.18 Функция NMIN

NMIN	- минимум из ряда чисел	c = nmin(a,b,…)

Пример: c=nmin({Рвх1},{Рвх2},0.7)

6.3.1.19 Функция NMAX

NMAX	- максимум из ряда чисел	c = nmax(a,b,…)

Пример: c=nmax(10,{T1},{T2},{T3})

6.3.1.20 Функция INT

INT	- взятие целой части	c = int(a)

6.3.1.21 Функция RESTDIV

RESTDIV	- взятие остатка от деления	c = restdiv(a,b)

6.3.2 Логические функции

6.3.2.1 Функция DOST

DOST	- определение признака достоверности выражения	c = dost(a) c=1, если а достоверно c=0, если а недостоверно

6.3.2.2 Функция TRUE

TRUE	- возвращаемое функцией значение всегда будет достоверным	c = true(a)

Если значение хотя бы одного из параметров, входящих в выражение на момент его использования в вычислениях было недостоверным, то значение рассчитываемого выражения также будет недостоверным. Это свойство вычислителя может быть обойдено при помощи использования функции ”**TRUE**”. Эта функция работает всегда правильно, если она распространяется на всю строку, т.е.:

Q = TRUE(A+B)	работает всегда правильно.
Q = C+TRUE (A+B)	не всегда (учитывается достоверность C).

6.3.2.3 Функция FALSE

FALSE	- возвращаемое функцией значение всегда будет недостоверно	c = false (a)

6.3.2.4 Функция GT

GT	- больше	c = GT(a,b), c = 1, если a > b

c = 0, если a < b
c = 0, если a == b

6.3.2.5 Функция GE

GE	- больше или равно	c = GE(a,b), c = 1, если a > b c = 0, если a < b c = 1, если a == b

6.3.2.6 Функция LT

LT	- меньше	c = LT(a,b), c = 0, если a > b c = 1, если a < b c = 0, если a == b

6.3.2.7 Функция LE

LE	- меньше или равно	c = LE(a,b), c = 0, если a > b c = 1, если a < b c = 1, если a == b

6.3.2.8 Функция EQ

EQ	- равно	c = EQ(a,b), c = 1, если a == b c = 0, если a != b

6.3.2.9 Функция NE

NE	- не равно	c = NE(a,b), c = 0, если a == b c = 1, если a != b

6.3.2.10 Функция NOT

NOT	-поразрядное инвертирование	c = NOT(a), c= 1010, если a = 0101

6.3.3 Функции работы с битами и байтами

6.3.3.1 Функция BIT

BIT	- получить значение бита из двойного слова.	c = bit(dw,bit0) dw – двойное слово; bit0 – номер бита (0..31). 0 – младший бит. Например: c = bit(101b,0) c равно 1; c = bit(101b,1) c равно 0.

6.3.3.2 Функция BITS

BITS	- получить значение группы битов двойного слова по маске mask, начиная с бита bit0. Младший бит 0.	c = bits(dw,bit0,mask) dw – двойное слово; bit0 – номер бита (0..31). 0 – младший бит. mask – битовая маска. Например: Выделить младшую вторую тетраду из двойного слова. c = bits (dw,4,0Fh)

6.3.3.3 Функция BXCHG

BXCHG	- переставить в байты в двойном слове. byteseq 4-х значная десятичная константа, задаёт последовательность байт в результирующем двойном слове. 1 – младший байт, 4 – старший байт в двойном слове. 0 – обнулить байт.	c = bxchg(dw,byteseq) Например: Переставить все байты в обратной последовательности. c = bxchg(44332211h,1234) Результат, с равно 11223344h.

6.3.4 Функции над временем

Функции астрономической даты и астрономического времени предназначены для работы с астрономическим временем, исчисляемым в секундах от 0 часов 0 минут 0 секунд 1-го января 1970 года.

6.3.4.1 Функция TIME

TIME	- получить текущее время в секундах от 00:00:00 01/01/1970 года;	c = time()

6.3.4.2 Функция SECOND

SECOND	- выделить секунды. Возвращаемое значение от 0 до 59	a = time() c = second(a)

6.3.4.3 Функция MINUTE

MINUTE	- выделить минуты. Возвращаемое значение от 0 до 59.	A = time() c = minute(a)

6.3.4.4 Функция HOUR

HOUR	- выделить часы.	A = time()

	Возвращаемое значение от 0 до 23.	c = hour(a)

6.3.4.5 Функция MONTHDAY

MONTHDAY	- выделить день в месяце. Возвращаемое значение от 1 до 31.	A = time() c = monthday(a)

6.3.4.6 Функция MONTH

MONTH	- выделить номер месяца. Возвращаемое значение от 0 до 11.	A = time() c = month(a)

6.3.4.7 Функция YEAR

YEAR	- выделить год. Возвращаемое значение полный, 4-х значный номер года (1970, 2010 и т.д.).	a = time() c = year(a)

6.3.4.8 Функция WEEKDAY

WEEKDAY	- получить номер дня в неделе. Возвращаемое значение 0 - Воскресенье, 6 - Суббота.	A = time() c = weekday(a)

6.3.4.9 Функция YEARDAY

YEARDAY	- получить номер дня в году. Возвращаемое значение от 0 до 365.	A = time() c c = yearday(a)

6.3.4.10 Функция MAKETIME

MAKETIME	- рассчитать время в секундах от 00:00:00 01/01/1970 года из пользовательских данных.	C = maketime(hour,min, sec,monthday,month,year), где hour -часы (0…23); min -минуты (0…59); sec -секунды (0…59); monthday - число месяца (1…31); month-месяц (0…11); year -год (1970…).

6.3.5 Функции времени выполнения

Эти функции позволяют оценить время выполнения цикла алгоритма.

6.3.5.1 Функция CYCLESEC

CYCLESEC	- получить период запуска алгоритма в секундах. Это величина указана в конфигурации алгоблока в тиках таймера.	A = cyclsec()

6.3.5.2 Функция EXECSEC

EXECSEC	- получить измеренное время выполнения алгоритма в секундах.	A = execsec()

6.3.6 Функции над таймерами

Функции для работы с таймерами (счётчиками времени) предназначены для работы с временем, насчитанным в параметрах типа “СЧЕТЧИК ВРЕМЕНИ” базы данных комплекса ЗОНД.

6.3.6.1 Функция TIMERMSEC

TIMERMSEC	- получить число миллисекунд насчитанных счётчиком времени. Возвращаемое значение от 0 до 999.	c = timermsec(a), где a - значение счётчика времени

6.3.6.2 Функция TIMERSEC

TIMERSEC	- получить число секунд насчитанных счётчиком времени. Возвращаемое значение от 0 до 59.	c = timersec(a), где a - значение счётчика времени

6.3.6.3 Функция TIMERMIN

TIMERMIN	- получить число минут насчитанных счётчиком времени. Возвращаемое значение от 0 до 59.	c = timermin(a), где a - значение счётчика времени

6.3.6.4 Функция TIMERHOUR

TIMERHOUR	- получить число часов насчитанных счётчиком времени.	c = timerhour(a), где a - значение счётчика времени

6.3.6.5 Функция MAKETIMER

MAKETIMER	. - рассчитать значение счётчика из пользовательских данных.	c = maketimer(hour, min, sec,msec), где hour -часы счётчика;

min -минуты (0..59);
sec -секунды (0..59);
msec -миллисекунды (0..999);

6.3.7 Функции счётчиков тиков

Функции этой группы позволяют получить доступ к счётчику прерываний от таймера (тиков).

6.3.7.1 Функция GETTICKS

GETTICKS	- получить число тиков таймера от момента запуска программы. Если значение аргумента отлично от нуля, функция возвращает разность между текущим значением счётчика тиков и агрументом.	c = getticks(prev_tick_cnt), где prev_tick_cnt - 0. Или предыдущее значение счётчика тиков таймера;

6.3.7.2 Функция TICKSIZE

TICKSIZE	- получить интервал тиков таймера в секундах.	c = ticksize ()

6.3.8 Функция счётчик секунд CURRENTSEC

CURRENTSEC	- получить число секунд, прошедшее с момента запуска программы на выполнение.	c = currentsec ()

6.3.9 Функции перезагрузки

Функции позволяют осуществить перезагрузку контроллера или компьютера, на котором функционирует программный модуль Зонд. Функции реализованы только в среде DOS, в WIN никаких действий не происходит.

6.3.9.1 Функция STOP_SOFTDOG

STOP_SOFTDOG	- перезагрузка, вызываемая срабатыванием программного сторожевого таймера потока алгоблока файловые операции	c = stop_softdog(),

	закрываются, образуется файл coredump.txt)	

6.3.9.2 Функция RESET

RESET	- мягкая перезагрузка, вызываемая искусственным делением на 0 (файловые операции закрываются, образуется файл coredump.txt, в регистре ax процесса задачи алгоблока значение из параметра функции).	c = reset(a), где a – значение параметра

6.3.9.3 Функция REBOOT

REBOOT	- мягкая перезагрузка (файловые операции закрываются).	c = reboot(),

6.3.10 Функции алгоритмов управления

6.3.10.1 Функция SET_WAIT

Функция работает аналогично оператору **SET**. Отличие состоит в том, что функция выполняется синхронно и возвращает результат выполнения команды.

SET_WAIT	- установить значение. Возвращаемое значение c: 0 - успех, заданное значение параметра достигнуто; 1 - тайм-аут; 2 - возникла ошибка	c = set_wait (sys,state,time_out), где sys - системный номер; state - устанавливаемое значение; time_out - время в секундах, отводимое на выполнение операции.

6.3.10.2 Функция SET_UNCOND

Функция работает аналогично оператору **SET**. Отличие состоит в том, что реальное воздействие на объект (в виде команд протокола) выполняется в любом случае, даже в том, когда параметр уже имеет требуемое состояние (например, команда открыть будет подана на открытый клапан). Функция удобна в применении, когда множество значений на запись в определённый адрес (адресация согласно протоколу, например, modbus) не совпадает с множеством значений на чтение из этого адреса.

SET_UNCOND	- установить значение. Возвращаемое значение	c = set_uncond (sys,state), где

не несёт информации	sys - системный номер; state - устанавливаемое значение;

6.3.10.3 Функция PIDREG

Функция предназначена для расчёта одного шага алгоритма ПИД-регулирования. Может применяться в задачах регулирования технологических процессов с постоянными времени от единиц секунд. Результатом расчёта (возвращаемым значением) является значение сигнала управления (воздействия на исполнительный механизм (ИМ)), которое должно быть подано на ИМ или на вход регулятора внутреннего контура явно (реализация функции не содержит команд управления).

Для исключения резкого изменения сигнала управления при ступенчатом изменении задания (удара) регулятор может иметь фильтр сигнала задания.

Для уменьшения шума сигнала обратной связи, связанного с его квантованием по времени и по уровню, регулятор может иметь фильтр сигнала обратной связи.

Регулятор использует для работы блок из 30 переменных алгоблока.

PIDREG - рассчитать значение сигнала управления. Возвращаемое значение - значение сигнала управления	u = pidreg (boi, S0_Ki,S1_Kp,S2_Kd, g,fu,y,y_diap,umin,umax,Tf, u_vmax,g_vmax, reg_mode,yf_size,yf_type,adapt_type, gf_type,def_type,y_trust),

Функция регулятора имеет следующие параметры

boi - номер первой переменной блока переменных алгоблока (с 1);

S0_Ki,S1_Kp,S2_Kd - И,П,Д коэффициенты регулятора (начальные значения, реальные значения могут меняться в результате применения алгоритмов адаптации);

g - задание на регулятор;

fu - реальное положение исполнительного механизма (если в БД такой сигнал отсутствует, можно использовать рассчитанный сигнал u;

y - сигнал обратной связи (значение регулируемой величины);

y_diap - диапазон сигнала обратной связи (используется только при адаптации);

u_min,u_max - ограничение на выходной сигнал управления ИМ;

Tf - постоянная времени входного фильтра (при использовании входного фильтра-инерционного звена);

u_vmax - ограничение по скорости сигнала управления ИМ (размерность сигнала {размерность сигнала управления}/с), при <=0 ограничение не происходит;

g_vmax - ограничение по скорости сигнала задания (размерность сигнала {размерность сигнала задания}/с, при использовании входного фильтра с ограничением скорости)

reg_mode - режим регулятора

0 - ручной,

1 - автоматический,

yf_size - размер буфера фильтра обратной связи (1-8)

yf_type - тип фильтра обратной связи
0 - нет фильтра
1 - медианный фильтр (ближайшее к среднему)
2 - среднее
adapt_type - тип адаптации
0 - нет адаптации, коэффициенты постоянны
1 - при условии fabs(e/y_diap) < 0.01 коэффициенты помножаются
s0 = s0 * 0.5;
s1 = s1 * 0.1;
s2 = s2 * 0.1;
gf_type тип фильтра сигнала задания
0 - нет фильтра
1 - ограничение по скорости (при ступеньке на входе сигнал нарастает монотонно)
2 - инерционное звено (при ступеньке на входе сигнал нарастает по экспоненте
def_type - использовать фильтр сигнала Д-канала
0 - не использовать (тип среднее, размер буфера 4)
1 - использовать
y_trust - достоверность сигнала обратной связи (1-достоверно, 0-нет)

Переменные блока распределены следующим образом:

boi+0 Число тиков с начала работы Зонд
boi+1 Рассчитанный такт регулятора, с
boi+2 сигнал задания после входного фильтра
boi+3 сигнал обратной связи после фильтра
boi+4 ошибка
boi+5 ошибка на предыдущем шаге
boi+6 S1_Кп, S0_Ки, S2_Кд - реальные значения коэффициентов
boi+9 значения каналов П,И,Д
boi+12 интегратор
boi+13 счётчик тактов при старте
boi+14 сигнал управления с учётом ограничения
boi+15 состояние регулятора
0 - нормальная работа в автоматическом режиме
1 - ручной режим
2 - старт (в автоматическом режиме первые yf_size тактов работы алгоблока для заполнения фильтра обратной связи)
3 - недостоверность сигнала обратной связи в автоматическом режиме
4 - некорректные (<0) значения коэффициентов S1_Кп, S0_Ки, S2_Кд
5 - некорректное (меньше такта) значение Tf
6 - некорректное (<0) значение g_vmax
7 - не выполнены необходимые условия
u_max>u_min, 1<yf_size<8, некорректные значения
gf_type, yf_type, adapt_type
Во всех состояниях, кроме 0, сигнал управления не вычисляется, а внутренние переменные регулятора поддерживаются в таком состоянии, чтобы минимизировать удар при переходе в состояние 0

boi+16 индекс в фильтре обратной связи
boi+17 индекс в фильтре Д-канала
boi+18 фильтр обратной связи
boi+26 фильтр Д-канала

Необходимым условием работы цифрового регулятора является точность выдержки такта регулятора (т.к. его значение входит в алгоритм расчёта) и выполнение теоремы Котельникова применительно к контуру управления - такт регулятора д.б. более чем в 10 раз быстрее, чем постоянная времени (в случае не первого порядка - максимальная) объекта. Один из способов настройки ПИД регулятора - на основе метода обратной задачи динамики.

Если считаем, что объект - инерционное звено первого порядка (K/(Тоб*s+1)), где Тоб - постоянная времени объекта (наше о нем начальное представление) и требования к замкнутой системе - тоже объект первого порядка, по динамике несколько быстрее разомкнутого объекта, тогда структура регулятора - ПИ (Кд=0). Для его коэффициентов должно выполняться Кп/Ки=Тоб. Если с сохранением этой пропорции Ки и Кп увеличивать, это равносильно желанию увеличивать К (коэффициент усиления) регулятора, уменьшать Тж - желаемого быстродействия замкнутой системы.

Например, при Тоб = 100 начинаем с Тж=0.5*Тоб=50, тогда Кп=2, Ки=0.02 соотв. более медленной замкнутой системе. При Тж=0.2*Тоб=20, Кп=5, Ки=0.05 - быстрее. Задача - выбрать максимально быстрый вариант, соответствующий более качественному управлению, что в реальной системе будет ограничено появлением участков вхождения сигнала управления при нормальных условиях работы в насыщение, либо излишне шумным для применяемого исполнительного механизма сигналом управления. В обоих случаях следствием будет ухудшение качества управления.

7. Выполнение алгоритма

Алгоритм выполняется кода показан на Рис. 7-1.

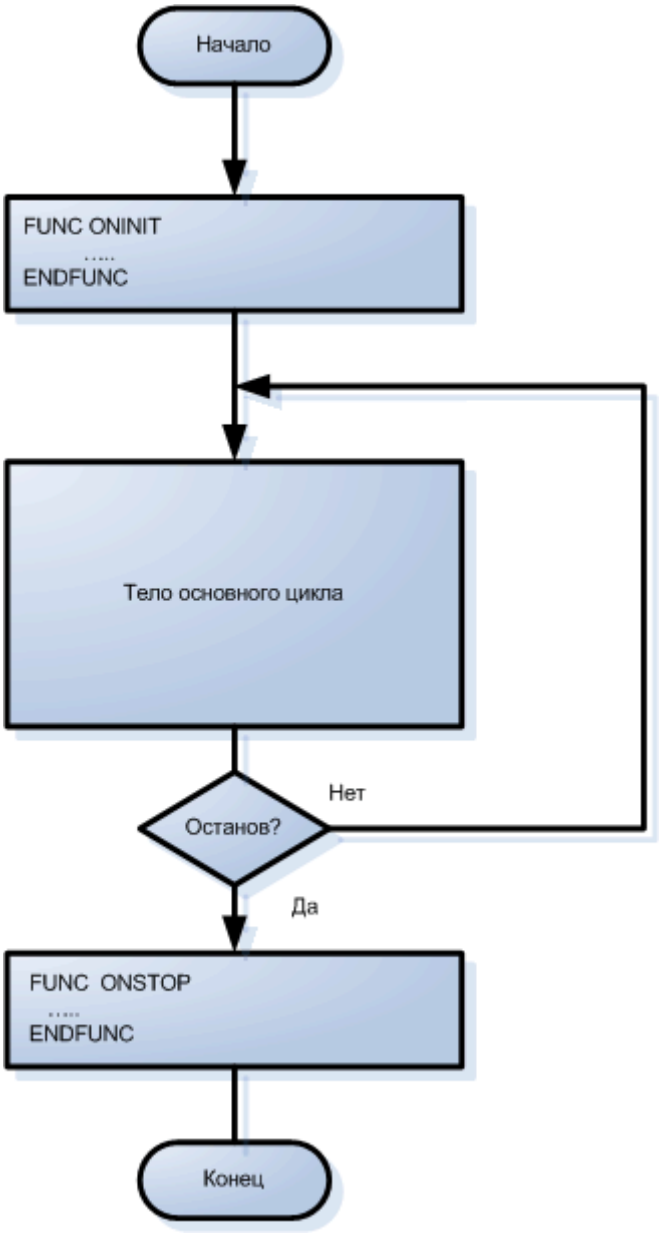


Рис. 7-1. Блок-схема выполнения алгоритма

Если создана пользовательская функция ONINIT, выполнение начинается с неё. Далее до получения, с заданным периодом выполняется основное тело алгоритма. Останов происходит по запросу пользователя и при завершении работы программы. Перед завершением, если объявлена, будет выполнена функция ONSTOP.

8. Панель инженера модуля “Вычислитель”

Панель инженера модуля «УСО Вычислитель» изображена на Рис. 8-1. Она состоит из нескольких частей:

- дерево параметров конфигурации алгоблоков;
- панель инструментов «Вычислителя»;
- окно текстового редактора;
- окно параметров базы данных;
- окно переменных и функций алгоблока;
- окно сообщений;
- служебное окно;

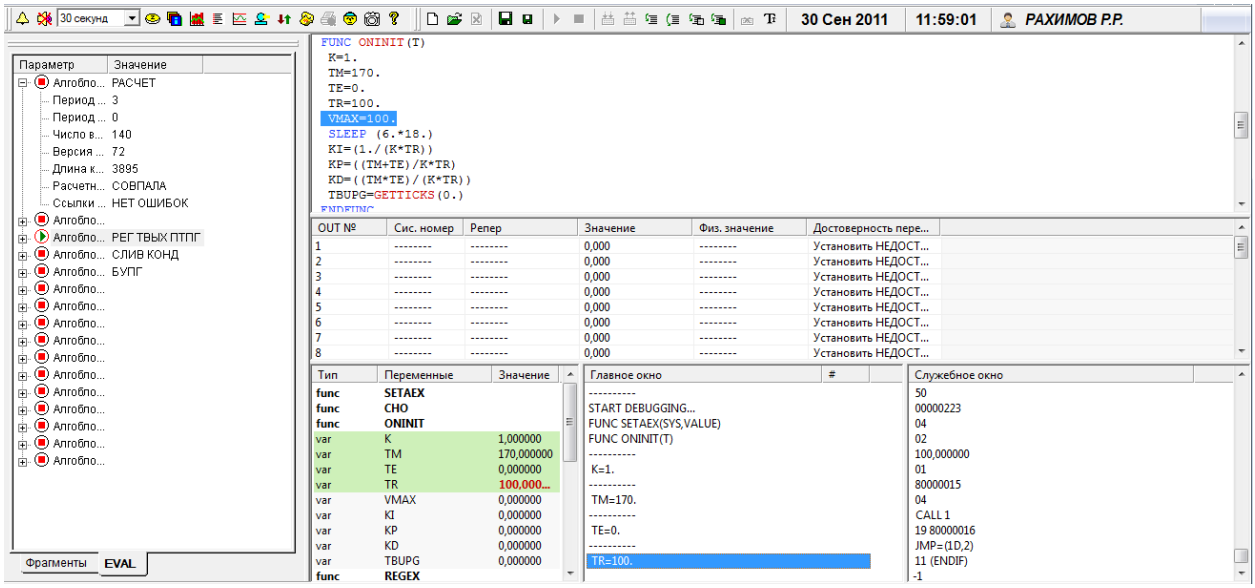


Рис. 8-1 Панель инженера модуля «Вычислитель»

8.1.1 Дерево параметров конфигурации алгоблоков

Дерево параметров содержит 16 веток – «Алгоблок». Элементы конфигурации алгоблока описаны ниже:

Столбец	Значение
Алгоблок	Название алгоблока
Период выполнения, тик	Период выполнения в единицах (тиках системного таймера) по 1/18 секунды
Период записи значения, мин	Период сохранения значений переменных в минутах;
Число выходных переменных, шт.	Количество выходных переменных алгоблока;
Версия кода	версия объектного (откомпилированного) кода;
Длина кода	длина кода в байтах;

Столбец	Значение
Расчётная CRC	результат проверки контрольной суммы кода (должно быть «СОВПАЛА»);
Ссылки на параметры БД	результат проверки ссылок на параметры базы данных (должно быть «Ok» или «НЕТ ОШИБОК»).

Выделенный в дереве параметром алгоблок называется текущим. Все операции производятся с текущим алгоблоком.

Версия объектного кода повышается при изменении структуры конфигурации УСО Вычислитель или при реализации новых конструкций языка (операторов, функций) при их наличии в тексте алгоблока (каждой конструкции сопоставлена максимальная версия конфигурации). Приложения комплекса Зонд определённых версий сборки способны создать код максимальной для себя версии, но реально создают исходя состава текста алгоблока. Выполнимые модули Зонд не выполняют алгоблок, если версия конфигурации Вычислителя больше максимальной для модуля данной версии сборки.

Алгоритм выполняется циклически с заданным периодом. Период задаётся в тиках системного таймера (55мс = 1/18 сек.). Если задан ноль, вычисления производятся непрерывно, по окончании расчёта сразу начинается новый цикл.

Значения расчётных переменных также сохраняются на диске с заданным периодом. Величина периода сохранения задаётся в минутах. Ноль в этом поле означает, что значения переменных сохранять на диске не нужно.

Количество выходных переменных подбирается в зависимости от потребностей алгоритма и не должно превышать 65535. В целях экономии оперативной памяти и объёма требуемого для хранения значений выходных переменных дискового пространства рекомендуем устанавливать это число равным номеру последней реально используемой переменной. Периоды и количество переменных задаются редактирование дерева параметров алгоблока.

При нажатии правой кнопки на дереве параметров произойдёт открытие всплывающего меню. Пункты меню совпадают со значениями соответствующих кнопок на панели инструментов УСО «Вычислитель» (см. раздел 8.1.2).

Возможно три состояния алгоблока, обозначенных соответствующим знаком:


Алгоблок запущен.


Алгоблок остановлен.


Пошаговая отладка алгоблока. Выполняется пошаговая отладка алгоблока.

8.1.2 Панель инструментов «Вычислитель» (Evaluator)

Панель инструментов «Вычислитель» представлена на Рис. 8-2.

Рис. 8-2 Панель инструментов «Вычислитель»

Набор кнопок панели позволяет выполнять следующие операции в УСО Вычислитель:

 *Новый файл.* Создаёт новый текстовый файл для написания кода алгоблока.

 *Открыть файл.*

 *Закрыть файл.*

 *Сохранить файл.*

 *Сохранить файл как.* Сохранить файл под новым именем.

 * *Запустить выполнение алгоблока.*

 * *Остановить выполнение алгоблока.*

 *Транслировать алгоблок.* Компилирует код в открытом документе.

 *Ретранслировать алгоблок.* Извлекает код из выделенного алгоблока.

 * *Шаг вперед.* Выполнить один шаг алгоритма. Если отладка выключена, то произойдёт трансляция открытого файла в алгоблок, а затем запустится режим отладки.

 * *Большой шаг.* Выполнить один шаг алгоритма. Если в текущем шаге вызов пользовательской функции, то она выполняется за один шаг.

 * *Включить останов.* После выполнения основного цикла алгоблока алгоритм начнёт выполнение функции OnStop, при её наличии.

 * *Остановить отладку.* Принудительная остановка отладки.

 *Очистить алгоблок.*

 *Настройки.* Открывает диалог настроек УСО Вычислителя.

*- данные кнопки неактивны в программе «Конфигуратор».
Их использование возможно в ПО «Зонд».

8.1.3 Горячие клавиши

При работе с УСО «Вычислитель» Вы можете воспользоваться следующими комбинациями клавиш:

Комбинация клавиш	Функция
<Ctrl>+<C> или <Ctrl>+<Ins>	Копировать текст из буфера обмена
<Ctrl>+<V> или <Shift>+<Ins>	Вставить текст из буфера обмена
<Ctrl>+<X> или	Вырезать текст в буфер обмена

<Shift>+<Delete>	
<Ctrl>+<Shift>+<S>	Сохранить файл под новым именем
<F2>	Свернуть окно «Текстовый редактор»
<F3>	Свернуть окно «База данных»
<F4>	Свернуть окна переменных, сообщений и служебное
<F7>	Транслировать алгоблок
<F8>	Ретранслировать алгоблок
<F9>	Шаг вперёд
<F10>	Большой шаг
<F11>	Включить останов
<F12>	Остановить отладку

8.1.4 Настройка УСО Вычислитель

Чтобы открыть окно настроек УСО «Вычислитель» нажмите кнопку «Настройки»  на панели управления «Вычислитель».

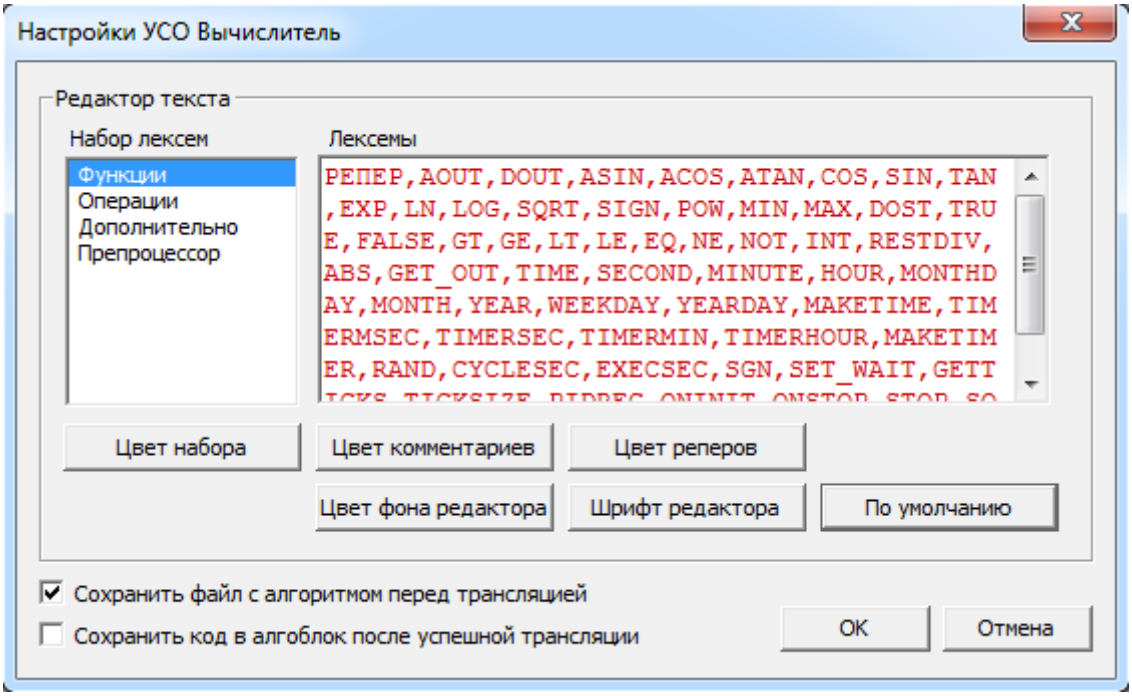


Рис. 8-3 Вид окна настроек УСО Вычислитель

В окне «Набор лексем» перечислены группы лексем, у которых можно менять цвет. Наборы лексем строятся автоматически, используя данные компилятора.

Доступные настройки:

- *Цвет набора*. Изменяет цвет слов, входящих в состав выделенного в окне набора лексем;
- *Цвет комментариев*. Изменяет цвет закомментированного текста.

- *Цвет реперов*. Изменяет цвет реперов, заключённых в фигурные скобки.
- *Цвет фона редактора.* Изменяет цвет фона редактора текста.
- *Шрифт редактора.* Изменяет параметры базового шрифта редактора. В том числе его цвет.
- *По умолчанию.* Нажатие этой кнопки сотрёт все сделанные в настройках изменения и вернёт их в значения «по умолчанию».
- *Сохранить файл с алгоритмом перед трансляцией.* Программа не будет транслировать «Новый файл.evl» без его сохранения на диск под новым именем.
- *Сохранить код в алгоблок после успешной трансляции.* Автоматически сохранит код в алгоблок, если трансляция завершена без ошибок.

Настройки сохраняются в реестре при закрытии УСО «Вычислитель».

8.1.5 Окно текстового редактора

При открытии УСО Вычислитель в редакторе находится пустой файл.

Вы можете:

- начать написание кода в окне;
- ретранслировать алгоритм из текущего алгоблока (кнопка «Ретранслировать»);
- открыть файл с алгоритмом (кнопка «Открыть файл»)
- выполнить настройку редактора (кнопка «Настройки»)

Текстовый редактор выполняет следующие функции:

- Подсветка функций, констант «УСО Вычислитель»;
- Подсветка реперов, заключённых в фигурные скобки;
- Окно «Вставка репера» при вводе символа «{»; сохранение его габаритов;
- Окно «Вставка служебных слов» при выборе пункта «Функции» в всплывающем меню; сохранение его габаритов;
- Всплывающая подсказка с описанием параметров и назначении функции при её написании и вводе символа “(“;
- Автоматическое дополнение пробелами при переводе строки.
- Замена табуляции пробелами.
- Настройка шрифта;
- Настройка цвета фона;
- Поддержка клавиатурных сокращений WIN & DOS для работы с текстом.

Всплывающее меню

Всплывающее меню вызывается нажатием правой кнопки мыши по полю редактора.

Действие пунктов меню:

- *Вырезать*. Вырезает в буфер обмена выделенный фрагмент текста;
- *Копировать*. Копирует в буфер обмена выделенный фрагмент текста;
- *Вставить*. Вставляет из буфера обмена текст;

- *Функции*. Открывает окно «Вставка служебных слов», для вставки функций, операторов и предопределённых констант вычислителя;

Всплывающее меню репера

При нажатии правой кнопки мыши на репере, заключённом в фигурные скобки, откроется всплывающее меню с пунктами:

- *Паспорт <название репера>*. При нажатии левой кнопки мыши по нему откроется редактор паспорта репера.
- *Установка значения <название репера>*. Доступно при наличие свойства телеуправления у репера.

При отсутствии репера в базе данных или английской раскладке клавиатуры название пункта будет равно «Репер не найден».

Внимание: Данное меню работает только в русской раскладке клавиатуры.

Всплывающее окно «Вставка служебных слов»

Всплывающее окно «Вставка служебных слов» изображено на Рис. 8-4.

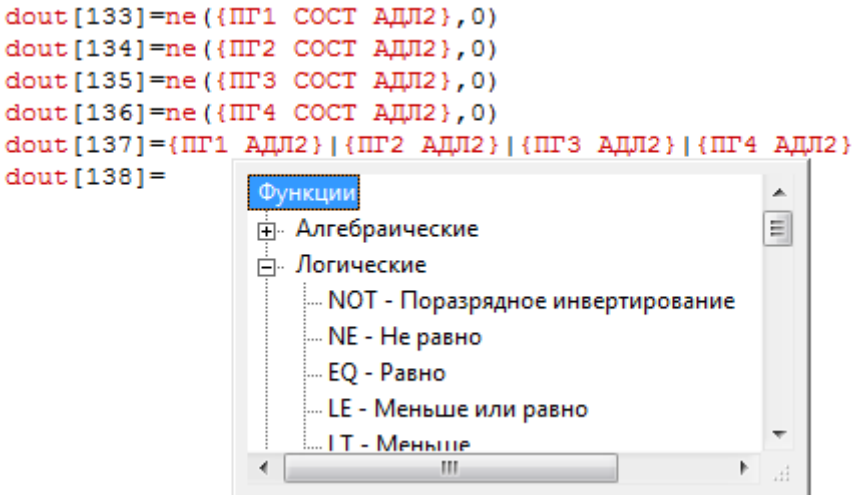


Рис. 8-4 Всплывающее окно "Вставка служебных слов"

Данное окно позволяет вставить в текущее место редактора функции, операции, операторы, предопределённые константы, директивы препроцессора.

Для этого раскройте соответствующий раздел дерева двойным нажатием левой кнопки мыши. Вставка слова осуществляется также двойным щелчком мыши.

Вы также можете работать с окно при помощи клавиатуры. При этом выбор пункта осуществляется кнопками «вверх», «вниз». Раскрытие раздела кнопкой «вправо» или «Enter». Добавление слова кнопкой «Enter». Закрытие окна кнопкой «Esc».

Всплывающее окно «Вставка репера»

Всплывающее окно «Вставка репера» изображено на Рис. 8-5.

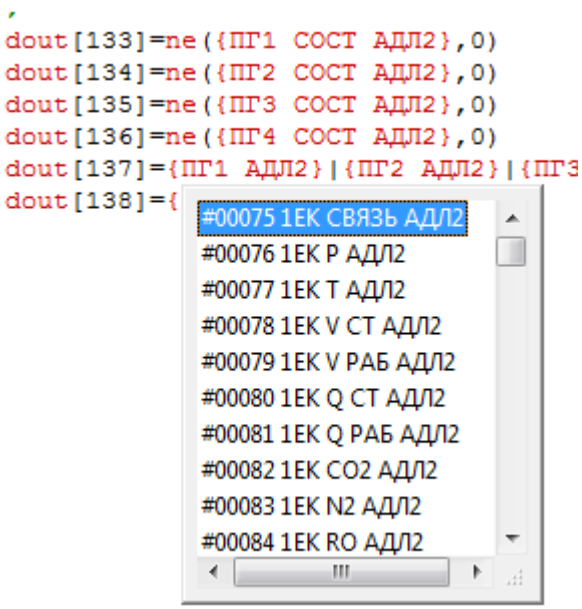


Рис. 8-5 Всплывающее окно "Вставка репера"

Для вызова окна «Вставка репера» нажмите кнопку «{». Для выбора репера Вы можете воспользоваться мышью или клавиатурой. Добавление репера в текст происходит по двойному нажатию левой кнопки мыши. Выбрать репер можно также с клавиатуры клавишами «вниз», «вверх» и последующим нажатием «Enter». Для выбора репера по системному номеру введите его номер.

Окно «Вставка репера» загружается один раз при первом использовании. При большом размере Базы Данных его первое появление может занять несколько секунд. При изменении структуры Базы Данных окно будет перестроено при первом вызове.

При написании встроенной функции и наборе скобки «(« отобразится всплывающая подсказка.

Если загруженный в редактор текст был подвергнут изменению, при выходе из редактора программа предложит сохранить текст в файл.

Редактор поддерживает одновременное открытие только одного файла.

8.1.6 Трансляция алгоблока

Для трансляции алгоблока нажмите кнопку «Транслировать» на панели управления.

Будет выполнено сохранение редактируемого файла на диск и его трансляция в алгоблок.

Перед выполнением трансляции программа проверит, открытый в редакторе файл новый или он открыт с диска.

Если в настройках УСО «Вычислитель» поднят флаг параметра *Сохранить файл с алгоритмом перед трансляцией*, то в случае трансляции нового файла, программа предложит его сохранить на диск.

Если трансляция завершена без ошибок, а в настройках поднят флаг параметра «*Сохранить код в алгоблок после успешной трансляции*», то программа

автоматически заменит код в алгоблоке на новый, иначе спросит пользователя о действии.

Если трансляция завершена с ошибками, то они будут перечислены в окне сообщений УСО «Вычислителя».

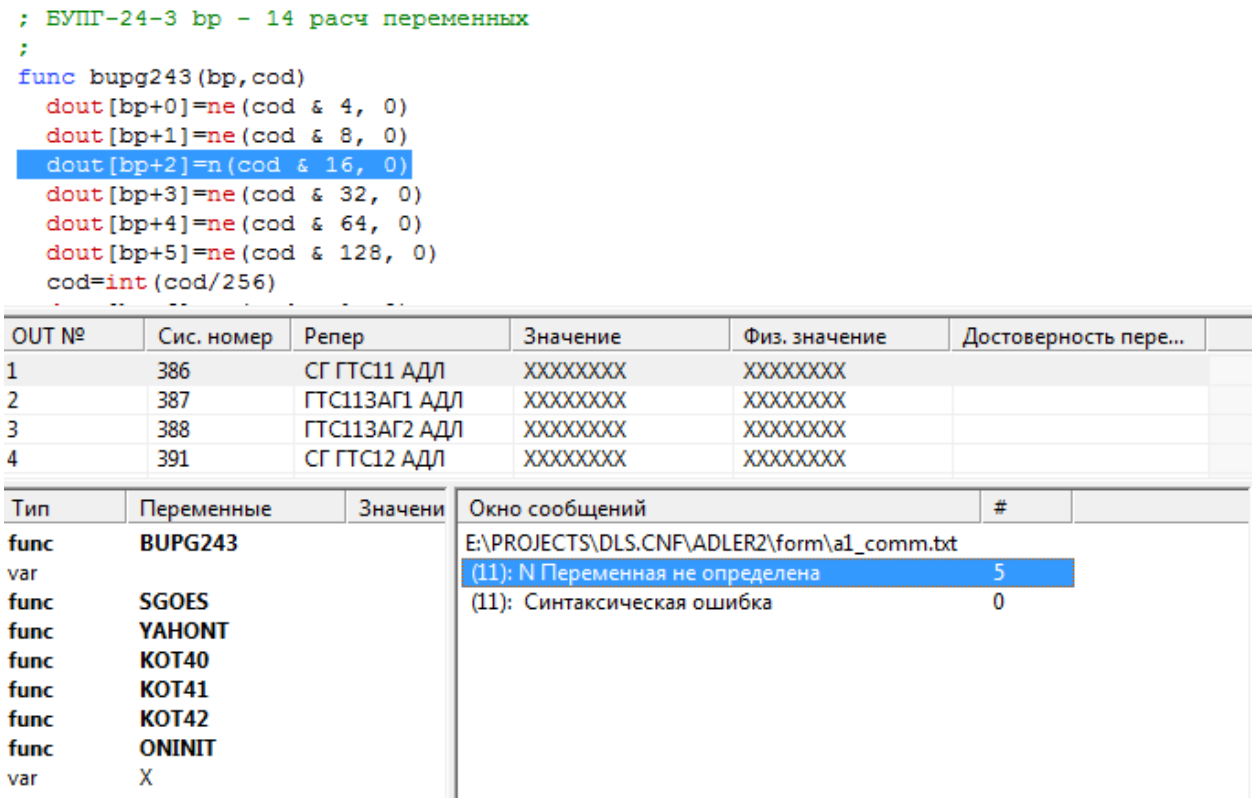


Рис. 8-6 Результат трансляции с ошибками.

Для просмотра строки с ошибкой нажмите два раза на строке с ошибкой в окне сообщений. Будет выделена строка с ошибкой в текстовом редакторе. Перечень сообщений об ошибках приведён в Разд.9.

В окне «Переменные и функции алгоблока» Вы увидите переменные и функции алгоблока. Столбец «Значение» будет заполнен для переменных при пошаговой отладке алгоритма.

8.1.7 Пошаговая отладка алгоритма

Данный режим доступен в ПО «Зонд» и не доступен в «Конфигураторе».

Для отладки программ возможен пошаговый режим выполнения объектного кода алгоблока. Для этого необходим корректно скомпилированный текст алгоритма в текстовом редакторе. Он может быть ретранслированным из алгоблока или скомпилированным из открытого файла.

Пошаговая отладка алгоритма реализуется следующими кнопками панели управления:

Шаг вперёд. Выполнить один шаг алгоритма. Если отладка выключена, то произойдёт сравнение алгоритма открытого файла с содержимым алгоблока. При наличии изменений будет предложено выполнить трансляцию алгоритма в

алгоблок. Если алгоритмы одинаковые, то запустится режим отладки.

 *Большой шаг.* Выполнить один шаг алгоритма. Если в текущем шаге вызов пользовательской функции, то она выполняется за один шаг.

 *Включить останов.* После выполнения основного цикла алгоблока алгоритм начнёт выполнение функции OnStop, при её наличии.

 *Остановить отладку.* Принудительная остановка отладки.

Для того чтобы запустить пошаговую отладку алгоритма необходимо нажать кнопку «Шаг вперёд» или «Большой шаг». Первое нажатие запустит трансляцию алгоритма из текстового редактора в текущий алгоблок. При успешной трансляции будет выполнен первый шаг алгоритма.

Выполняемая на данном шаге строка выделяется в текстовом редакторе.

Алгоритм выполнения программы описан в Разд. 7.

Если текущий шаг алгоритма содержит пользовательскую функцию, то следующим шагом будет пошагово выполняться код внутри функции. После возврата из функции следующим шагом будет следующая за вызовом функции строка.

Если Вы хотите выполнить пользовательскую функцию за один шаг, то нажмите кнопку «Большой шаг». Функция выполнится за один шаг и алгоритм перейдёт на следующую за вызовом функции строку.

Кнопки «Шаг вперёд» и «Большой шаг» эквивалентны для всех операций не связанных с вызовом пользовательской функции.

Особенности пошаговой отладки

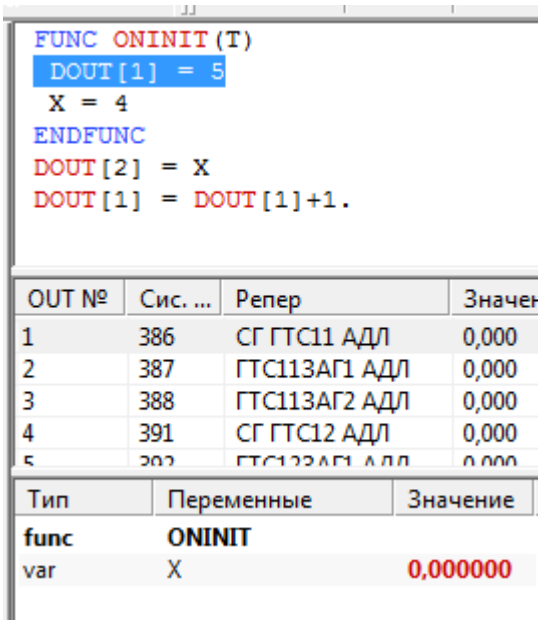
Завершение выполнения функции OnInit, OnStop, а также цикла основного алгоритма является отдельным шагом. В этот момент выделяется последняя строка в текстовом редакторе.

Выполнение основного цикла начинается с выделения первой строки в текстовом редакторе. Необходимо сделать ещё один шаг.

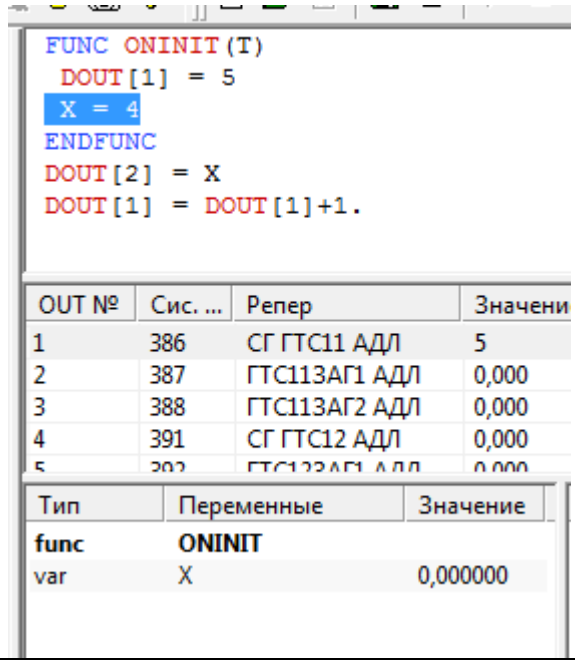
8.1.8 Пример выполнения алгоритма по шагам

Ниже представлен пример выполнения алгоритма по шагам.

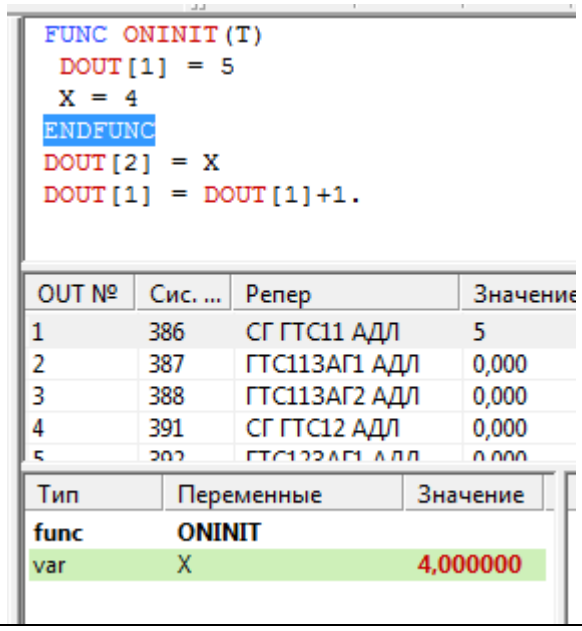
Шаг 1. Функция OnInit присутствует в алгоритме. Переходим к выполнению первой строчки ее кода. Первая команда алгоритма обнуляет переменные алгоблока.	Шаг 2. DOUT[1] = 5 выполнено. Переходим ко второй строчке.



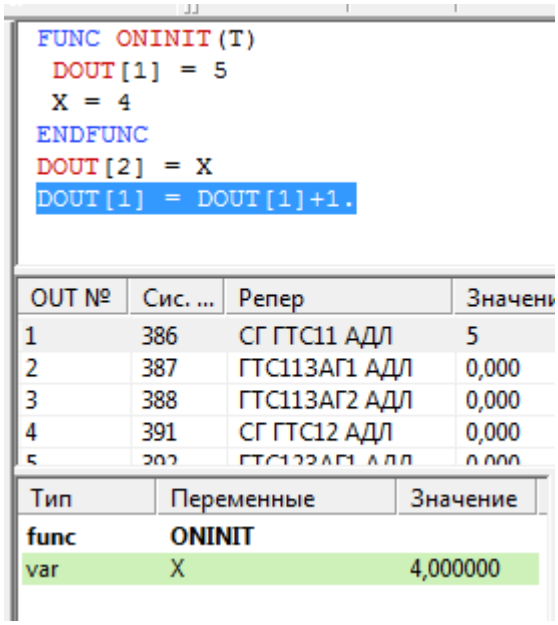
Шаг 3. Выполнено X=4. Значение переменной изменилось. Значение достоверно. Маркер стоит на завершении OnInit.



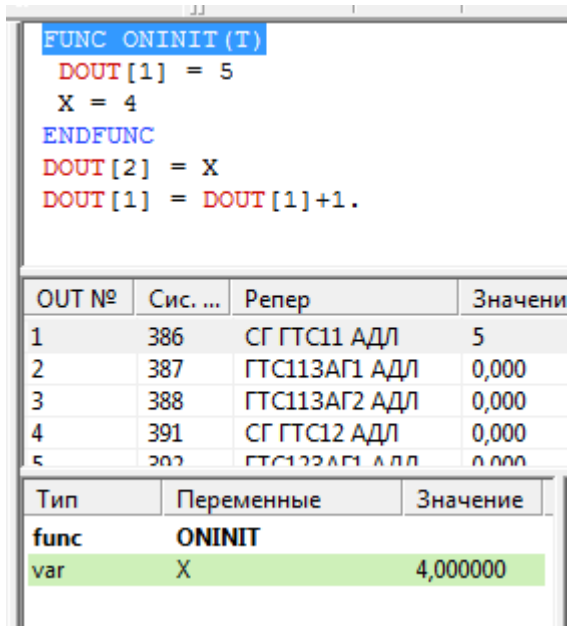
Шаг 4. Функция OnInit завершает своё выполнение. Выделяется последняя строка кода.



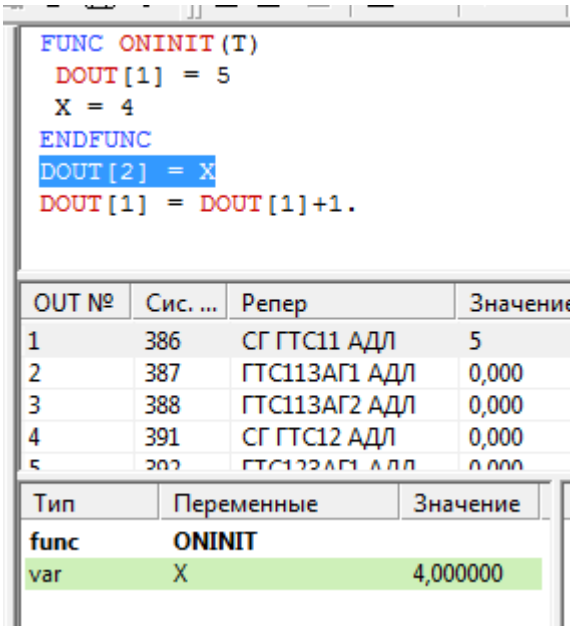
Шаг 5. Маркер переходит к выполнению основного кода программы. Выделена первая строка текста программы.



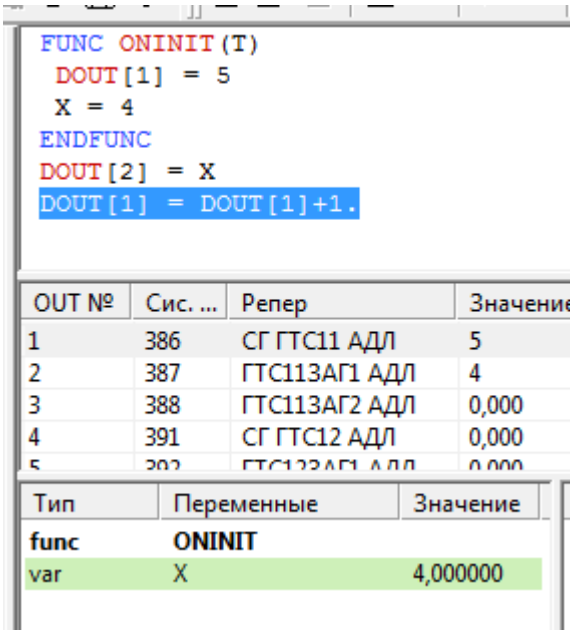
Шаг 6. Переходим к выполнению первой строки основного кода



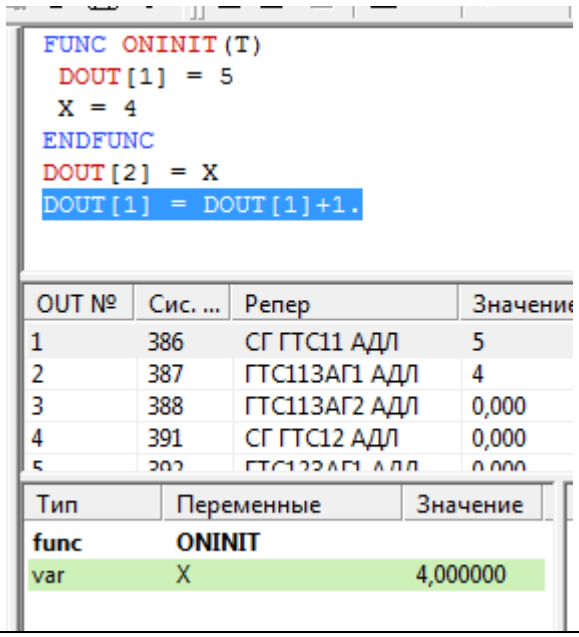
Шаг 7. Выполняется вторая строка основного кода.



Шаг 8. Завершение выполнения основного кода программы. Выделена последняя строка программы.



Шаг 9 равен Шагу 5.



Шаг 10 равен Шагу 6.. и так далее.

Для остановки выполнения программы на текущем шаге нажмите кнопку «Остановить отладку»  .

Если Вы хотите, чтобы программа остановилась после выполнения основного цикла, то нажмите кнопку «Включить останов»  . После выполнения последней строчки основного цикла программа перейдёт к выполнению функции OnStop, при ее наличии.

Внимание! Перед закрытием окна УСО Вычислитель необходимо завершить отладку всех алгоблоков.

9. Перечень сообщений об ошибках

9.1 Сообщение транслятора

Во время трансляции файла с исходным текстом формул могут возникать ошибки. Перечень сообщений об ошибках, а также возможные причины их возникновения приведены в Таблица 9-1.

Таблица 9-1

Сообщение	Причина возникновения
Синтаксическая ошибка	1. Лексема в указанной строке не является ни константой, ни переменной, ни индексированной переменной 2. Лексема не является операцией 3. Предполагается конец строки, однако она продолжается
Нет закрывающей скобки	1. Нет закрывающей ‘)’ в выражении со скобками или при вызове функции 2. Нет закрывающей ‘]’ в индексированной переменной
Это не выражение	Выражение состоит только из признака конца строки
Предполагается символ равенства	1. В строке без операторов не содержится операции присваивания 2. Операция присваивания находится в неположенном месте строки
Не переменная	Слева от операции присваивания не переменная
Переменная не определена	1. Имя переменной отсутствует в списке внутренних переменных, не совпадает ни с одним репером параметров в базе данных и не является выходной переменной 2. Номер выходной переменной превышает допустимый
Имя переменной обрезано	Имя внутренней переменной длиннее 15 символов, в таблица переменных заносится укороченное имя
Переполнена таблица переменных	Число временных переменных превышает допустимое
Неверное число аргументов	Число фактических аргументов функции не совпадает с числом формальных аргументов
Недопустимое значение индекса	Недопустимый индекс в индексированной переменной
Переполнен буфер объектного кода	Размер объектного кода программы превышает допустимый
Недопустимы вложенные функции	Начало объявления новой функции пользователя появилось раньше оператора ENDFUNC

Сообщение	Причина возникновения
Нет начала функции	1. Встречен оператор ENDFUNC без оператора FUNC 2. Встречен оператор RETURN без оператора FUNC
Нет конца функции	В программе нет оператора ENDFUNC
Переполнена таблица функций	Попытка объявления более чем 32 функций пользователя в одной программе
Предполагается скобка	Отсутствует открывающая ‘(’ в объявлении функции пользователя
Нет ENDIF	В программе нет оператора ENDIF
Лишний ENDIF	
Лишний ELSE	
Превышено количество IF	
Слишком длинная исходная строка	
Повторно объявлена ф-ция OnInit	
Повторно объявлена ф-ция OnStop	
Ошибка открытия файла	Указанный в #include файл не открывается
Файл уже добавлен ранее	Этот файл уже был добавлен #include
Длинный путь к файлу	Путь к файлу превышает макс. длину
Preprocessor buffer is full	Буфер препроцессора закончился. Уменьшите длину пути до файла.

9.2 Сообщения вычислителя алгоблока

При выполнении кода формулы могут возникать ошибки. Перечень этих ошибок, а также возможные причины их возникновения приведены в Таблица 9-2.

Таблица 9-2.

Код	Сообщение	Причина возникновения
1	Переполнение стека	Переполнение стека вычислителя. 1. Ошибка в коде формулы. 2. Внутренняя ошибка выполнения кода.
2	Стек пустой	Перевыборка стека вычислителя. 1. Ошибка в коде формулы. 2. Внутренняя ошибка выполнения кода.
3	Нет символьного файла	Для пошагового выполнения в отладчике необходим файл исходного кода
4	Ошибка в типе индексированной переменной	Переменная данного типа не может быть индексирована. 1. Ошибка в коде формулы. 2. Внутренняя ошибка выполнения кода. 3. Предусмотренный для функции блок переменных выходит за границу, определяемую числом переменных алгоблока
5	Ошибка в коде операции	Во время выполнения кода обнаружен недопустимый код операции. 1. Возможно несоответствие версий. Код создан

Код	Сообщение	Причина возникновения
		более поздней версией транслятора чем ретранслятор. 2. Код формулы разрушен (испорчен).
6	Присвоение значения невозможно	Попытка присвоения значения не переменной. 1. Ошибка в коде формулы. 2. Внутренняя ошибка выполнения кода.
7	Нет конца объектного кода	Не обнаружен признак конца объектного кода. Возможно код формулы повреждён (испорчен).
8	Операция с некорректным номером переменной	Значение индекса переменной лежит за допустимым диапазоном. 1. Ошибка в коде формулы. 2. Внутренняя ошибка выполнения кода.
9	Неверный системный номер	Ссылка на несуществующий в БД системный номер.
10	Неизвестная версия	Неизвестная версия двоичного кода. Возможно, нужно обновить ПО
11	Неожиданный return	Передача управления на некорректный адрес при выходе из тела функции
12	Недопустимое значение в set	Попытка установить недопустимое значение или параметр БД не допускает установку значения.
13	Ошибка выполнения сценария execute	Отсутствует файл сценария, или исходный текст сценария содержит ошибки
	Неизвестный код ошибки	Получен неопределённый код ошибки

9.3 Сообщения ретранслятора

Во время ретрансляции исходного кода формул могут выдаваться сообщения об ошибках, приведённые в Таблица 9-3.

Таблица 9-3.

Сообщение	Причина возникновения
Недопустимая операция	Во время разбора кода обнаружен недопустимый код операции. 1. Возможно несоответствие версий. Код создан более поздней версией транслятора чем ретранслятор. 2. Код формулы разрушен (испорчен).

9.4 Сообщения при запуске алгоблока на выполнение

Во время запуска на выполнение кода алгоблока в окно технологических сообщений (см.) могут выдаваться сообщения об ошибках, приведённые в Таблица 9-4.

Таблица 9-4.

Сообщение	Причина возникновения
АЛГОБЛОК #: ОШИБКА КС	Ошибка контрольной суммы кода алгоблока #. Возможно испорчен файл **eval.res**.
АЛГОБЛОК #: ОШИБКА ССЫЛКИ НА ПАРАМЕТР БД	В коде формулы содержаться ссылки на параметры БД, которые в данный момент отсутствуют. Возможно изменилось место расположение параметров в БД или они удалены.

10. Список используемых документов

Док. 1. Комплекс Программ «Зонд». Установка конфигурирование и запуск.

Док. 2. Комплекс Программ «Зонд». «Зонд2006» описание применения.

Док. 3. Комплекс программ «ЗОНД». Сообщения.

Док. 4. Комплекс программ «ЗОНД». Программа «Конфигуратор Базы Данных». Описание применения

11. Приложения

11.1 Алфавитный указатель ключевых слов

ABS 34
ACOS 34
ALL_IN_BASE 12
AOUTXXXXX 13
ASIN 34
ATAN 34
BEEP 24
COS 35
CURRENT_KVIT 12, 17, 27
CURRENTSEC 40
CYCLESEC 39
DELTA 16
DOST 36
DOUTXXXXX 13
ELSE 29, 30
ELSE-IF 30
ENDFUNC 33
ENDIF 30
EQ 37
EXECSEC 39
EXECUTE 24
EXP 35
FALSE 36
FLOW_MONTH 16
FLOW_PREVDAY 16
FLOW_TODAY_FALSE 16
FLOW_TODAY_TOTAL 16
FLOW_TODAY_TRUE 16
FUNC 31
GE 37
GETTICKS 40
GT 37
HOUR 38
IF 29, 30
IF-ELSE 29
INCLUDE 10
INITOUTS 13, 24
INT 36
LE 37
LN 35
LOCK_ARCH 27
LOG 35
LT 37
MAKETIME 38
MAKETIMER 40
MAX 35
MESREPER 23
MESSAGE 22
MIN 35
MINUTE 38
MONTH 38
MONTHDAY 38
NAG 12, 16
NE 37
NMAX 36
NMIN 36
NNG 16
NOT 37
NTG 12, 16
NVG 12, 16
ONINIT 34, 45
ONSTOP 34, 45
PIDREG 42
POW 35
RAND 35
REBOOT 41
RESET 41
RESTDIV 36
RETURN 33
SCALE 16
SCALEB 16
SECOND 38
SET 25, 41, 42
SET_STAT 26
SET_UNCOND 42
SET_UST 27
SET_WAIT 41
SGN 35
SIGN 35
SIN 35
SIREN_OFF 24
SIREN_ON 24
SLEEP 24
SQRT 35
STATUS_ALARM 12, 17, 27
STATUS_CAPTURE 12, 17, 26
STATUS_EQUIP 12, 17, 27
STATUS_KVIT_BAD 12, 17, 26

STATUS_KVIT_GOOD..... 12, 16, 26
STATUS_MESWIN 12, 16, 26
STATUS_PRFILE..... 12, 16, 26
STATUS_PRN 12, 16, 26
STATUS_SIREN 12, 17, 26
STATUS_TREATMENT..... 12, 17, 26
STOP_SOFTDOG 41
SYS_NUM.....16
TAN.....35
TICKSIZE.....40
TIME 37
TIMERHOUR.....40
TIMERMIN..... 39
TIMERMSEC 39
TIMERSEC..... 39
TRUE..... 36
UNLOCK_ARCH..... 27
VAG.....12, 16
VTG.....12, 16
VVG.....12, 16
WEEKDAY..... 38
YEAR..... 38
YEARDAY 38

11.1 Пример алгоритма

Ниже приводится пример расчёта с использованием функций. Формулы написаны для системы слежения за утечками в резервуарном парке (РТС - резервуар тонкостенный стальной).

```
; Расчет объема резервуара ------------------------------------------------
; L - текущий уровень
; Lmin - нижняя граница датчика
; L12  - точка перелома
; ПНД  - поправка на ....
; V - объем на предыдущем цикле вычислений
; K12,K11,K10 - коэф. на первом участке
; K22,K21,K20 - коэф. на втором участке
;-------------------------------------------------------------------------------
func ОБЪЕМ_РТС (V,ПНД,L,Lmin,L12,K12,K11,K10,K22,K21,K20)
 if ge(L,Lmin)       ; уровень в диапазоне измерения
  if le(L,L12)       ; выбираем участок
    K2=K12
    K1=K11
    K0=K10
  else
    K2=K22
    K1=K21
    K0=K20
  endif

  V=K2*L^2+K1*L+K0+ПНД
 else
  V=false(V)
 endif

 return (V)
endfunc

; Расчет состояния резервуара ----------------------------------------------
; i - смещение в массиве AOUT и DOUT
; NOM - номер РТС
; РЕЖИМ_РТС - 0-работа или 1-отключен
; N - режим 0-инициализация, 1-хранение, 2-прием/отпуск, 3 - отключение канала
; L_РУЛЕТ_РТС - уровень по рулетке
; L_РТС - уровень по датчику
; Lmin - нижний уровень измерения датчика
; L12 - точка разграничения участков
; K12,K11,K10 - коэф. для вычисления объема РТС на первом участке
; K22,K21,K20 - коэф. для вычисления объема РТС на втором участке
```

```
; Т_РУЧ_РТС - температура, замеренная вручную
; Т_РТС - температура по датчику
; ПЛОТН20_РТС - плотность продукта
; V_MAX - максимальный объем хранения
; L_ТЕК_РТС_VAG - предел уровня (ПДУ)
; М_ИЗМ_РТС_VAG - предел изменения массы
; Диапазон индексов 0 - 18
;-------------------------------------------------------------------------------
КП2=-0.00007003             ; коэф. для вычисления плотности
КП1=-0.001212
КП0=0.001786

funcРАСЧ_РЕЖ_РТС (i,NOM,РЕЖИМ_РТС,N,L_РУЛЕТ_РТС,ПНД_РТС,L_РТС,Lmin,L12,K12,K11,
K10,K22,K21,K20,Т_РУЧ_РТС,Т_РТС,ПЛОТН20_РТС,V_MAX,
L_ТЕК_РТС_VAG,М_ИЗМ_РТС_VAG,М_ИЗМ20_РТС,ДОП_УТС_РТС)

KPL=ПЛОТН20_РТС^2*КП2+ПЛОТН20_РТС*КП1+КП0

RET = 0;                    ; возвращаемое значение (признак печати)

if eq(РЕЖИМ_РТС,0)          ; РТС в работе
 if eq(N,0)                  ; инициализация ----------------------------
  AOUT[i]=L_РУЛЕТ_РТС-L_РТС   ; L_НАЧ_СЧ_РТС
                    ; V_НАЧ_РТС
  AOUT[i+1]=ОБЪЕМ_РТС (AOUT[i+1],ПНД_РТС,L_РУЛЕТ_РТС,
                       Lmin,L12,K12,K11,K10,K22,K21,K20)
  AOUT[i+2]=Т_РУЧ_РТС-Т_РТС   ; Т_ПОПР_РТС

  PL=ПЛОТН20_РТС*(1+KPL*(20-Т_РУЧ_РТС))
  AOUT[i+3]=AOUT[i+1]*PL      ; М_НАЧ_РТС

  ЦВЕТ=0                     ; ЦВЕТ_L_РТС = НОРМА
 else                        ; -------------------------------------------
  if (eq(N,1) | eq(N,2))     ; хранение или прием продукта
   AOUT[i+4]=L_РТС+AOUT[i]   ; L_ТЕК_РТС
                   ; V_ТЕК_РТС
   AOUT[i+5]=ОБЪЕМ_РТС (AOUT[i+5],ПНД_РТС,AOUT[i+4],
Lmin,L12,K12,K11,K10,K22,K21,K20)
   AOUT[i+6]=Т_РТС+AOUT[i+2] ; Т_ТЕК_РТС

   AOUT[i+7]=ПЛОТН20_РТС*(1+KPL*(20-AOUT[i+6]))   ; ПЛ_ТЕК_РТС
   AOUT[i+8]=AOUT[i+5]*AOUT[i+7]                  ; М_ТЕК_РТС

   AOUT[i+9]=AOUT[i+4]-L_РУЛЕТ_РТС                ; L_ИЗМ_РТС
   AOUT[i+10]=AOUT[i+5]-AOUT[i+1]                 ; V_ИЗМ_РТС
   AOUT[i+11]=V_MAX-AOUT[i+5]                     ; V_СВОБ_РТС
   AOUT[i+19]=AOUT[i+8]-AOUT[i+3]                 ; М_ЗНК_РТС со знаком

   if ge(AOUT[i+4],L_ТЕК_РТС_VAG)   ; Контроль ПДУ --------------------
    ЦВЕТ=1                    ; ЦВЕТ_L_РТС = ПДУ
    RET=1                     ; печать

    if eq((DOUT[i+13] & 1),0)      ; еще не было срабатывания
     AOUT[i+18]=time()             ; ТМ_20_РТСxxx - утечка/переток,...20,ПДУ
     DOUT[i+13]=1          ; засечка времени от ПДУ
    endif

   else
    if eq(N,1)                    ; режим хранения
     AOUT[i+12]=abs(AOUT[i+19])      ; М_ИЗМ_РТС

     if ge(AOUT[i+12],М_ИЗМ_РТС_VAG)  ; контроль по массе ---------
      RET=1

      if sign(AOUT[i+19])
       ЦВЕТ=3                     ; ЦВЕТ_L_РТС = УТЕЧКА
```

```
      else
       ЦВЕТ=2                    ; ЦВЕТ_L_РТС = ПЕРЕТОК
      endif

      if eq((DOUT[i+13] & 4),0)
       AOUT[i+18]=time()  ; ТМ_20_РТСxxx - утечка/переток,...20,ПДУ
       DOUT[i+13]=4               ; засечка времени утечка/переток

       if sign(AOUT[i+19])
        message "РТС %3.0f УТЕЧКА",NOM
       else
        message "РТС %3.0f ПЕРЕТОК",NOM
       endif
      endif
     else
                    ; контроль утечки/перетока 20 минут (по скорости)
      if ge(abs(М_ИЗМ20_РТС),ДОП_УТС_РТС) ;-------------------------
       RET=1

       if sign(М_ИЗМ20_РТС)
        ЦВЕТ=3          ; ЦВЕТ_L_РТС = УТЕЧКА
       else
        ЦВЕТ=2          ; ЦВЕТ_L_РТС = ПЕРЕТОК
       endif

       if eq((DOUT[i+13] & 2),0)   ; отсечка времени еще не сделана
        AOUT[i+18]=time() ; ТМ_20_РТСxxx - утечка/переток,...20,ПДУ
        DOUT[i+13]=2      ; засечка времени утечка/переток 20 мин.

        if sign(М_ИЗМ20_РТС)
         message "РТС %3.0f УТЕЧКА (ПО СКОРОСТИ)",NOM
         siren_on
        else
         message "РТС %3.0f ПЕРЕТОК (ПО СКОРОСТИ)",NOM
         siren_on
        endif
       endif
      else               ; контроль утечки по массе
       ЦВЕТ=0             ; ЦВЕТ_L_РТС = НОРМА
       AOUT[i+18]=0         ; время начала сигнала
       DOUT[i+13]=0         ; флаги
      endif
     endif
    else
     ЦВЕТ=0               ; ЦВЕТ_L_РТС = НОРМА - прием продукта
     AOUT[i+18]=0           ; время начала сигнала
     DOUT[i+13]=0           ; флаги
    endif
   endif
  else      ;----- режим ОТКЛЮЧЕН (все расчетные недостоверные) -----
   AOUT[i+4]=false(AOUT[i+4])       ; L_ТЕК_РТС
   AOUT[i+5]=false(AOUT[i+5])       ; V_ТЕК_РТС
   AOUT[i+6]=false(AOUT[i+6])       ; T_ТЕК_РТС
   AOUT[i+7]=false(AOUT[i+7])       ; ПЛ_ТЕК_РТС
   AOUT[i+8]=false(AOUT[i+8])       ; М_ТЕК_РТС
   AOUT[i+9]=false(AOUT[i+9])       ; L_ИЗМ_РТС
   AOUT[i+10]=false(AOUT[i+10])     ; V_ИЗМ_РТС
   AOUT[i+11]=false(AOUT[i+11])     ; V_СВОБ_РТС
   AOUT[i+12]=false(AOUT[i+12])     ; М_ИЗМ_РТС

   DOUT[i+14]=false(DOUT[i+14])     ; ЦВЕТ_РТС
   AOUT[i+15]=false(AOUT[i+15])     ; ШКАЛА_РТС
   AOUT[i+16]=false(AOUT[i+16])     ; НИЗ_РТС
   AOUT[i+17]=false(AOUT[i+17])     ; ВЕРХ_РТС
   AOUT[i+18]=0
   AOUT[i+19]=false(AOUT[i+19])     ; М_ЗНК_РТС со знаком
```

```
    ЦВЕТ=false(ЦВЕТ)
   endif
  endif

  if ((!DOST(L_РТС)) | (!DOST(T_РТС)))
   ЦВЕТ=false(ЦВЕТ)
  endif

  DOUT[i+14] = ЦВЕТ                ; ЦВЕТ_L_РТС
                                 ; расчет шкалы 100 мм
  L=AOUT[i+4]*1000                 ; L_ТЕК_РТС
  AOUT[i+15]=(L-int(L/100)*100)/1000   ; ШКАЛА_РТС
                                ; показания шкал
  AOUT[i+16]=(int(L/100)*100)/1000     ; НИЗ_РТС
  AOUT[i+17]=(int(L/100)*100+100)/1000  ; ВЕРХ_РТС

  if (RET)                         ; проверка выводить ли на печать
   if (AOUT[i+18])                   ; время != 0
    if eq(RESTDIV(minute(time()-AOUT[i+18]),20),0) ; и кратно 20 мин. от начала
     if eq((DOUT[i+13] & 8),0)
      RET=1
      DOUT[i+13]=DOUT[i+13] | 8
     else
      RET=0
     endif
    else
     DOUT[i+13]=DOUT[i+13] & NOT(8)
     RET=0
    endif
   endif
  endif

else    ;---- Всем признаки недостоверности (РТС в резерве) ----
  AOUT[i]=false(0)                 ; L_НАЧ_СЧ_РТС
  AOUT[i+1]=false(0)               ; V_НАЧ_РТС
  AOUT[i+2]=false(0)               ; T_ПОПР_РТС
  AOUT[i+3]=false(0)               ; M_НАЧ_РТС
  AOUT[i+4]=false(0)               ; L_ТЕК_РТС

  AOUT[i+5]=false(0)               ; V_ТЕК_РТС
  AOUT[i+6]=false(0)               ; T_ТЕК_РТС
  AOUT[i+7]=false(0)               ; ПЛ_ТЕК_РТС
  AOUT[i+8]=false(0)               ; M_ТЕК_РТС
  AOUT[i+9]=false(0)               ; L_ИЗМ_РТС
  AOUT[i+10]=false(0)              ; V_ИЗМ_РТС
  AOUT[i+11]=false(0)              ; V_СВОБ_РТС
  AOUT[i+12]=false(0)              ; M_ИЗМ_РТС
  DOUT[i+13]=0                     ; DOUT[i+13] - флаг против повторного срабат.
  DOUT[i+14]=false(0)              ; ЦВЕТ_РТС
  AOUT[i+15]=false(0)              ; ШКАЛА_РТС
  AOUT[i+16]=false(0)              ; НИЗ_РТС
  AOUT[i+17]=false(0)              ; ВЕРХ_РТС
  AOUT[i+18]=0
  AOUT[i+19]=false(0)              ; M_ЗНК_РТС со знаком

endif

;if (RET)
; message "RET=%1.0f РТС %3.0f",RET,NOM
;endif

return (RET)
endfunc

;------------------------ Расчет режима резервуаров ------------------------
```

```
 if (РАСЧ_РЕЖ_РТС (1,1,РЕЖИМ_РТС1,N1,L_РУЛЕТ_РТС1,ПНД_РТС1,L_РТС1,
4.7,5.9,-42.486,1548,-2859,-194.713,3234,-7505,T_РУЧ_РТС1,T_РТС1,
ПЛОТН20_РТС1,V_МАКС_РТС1,L_ТЕК_РТС1[VAG],M_ИЗМ_РТС1[VAG],
M_ИЗМ20_РТС1,ДОП_УТС_РТС1))
   sleep(5)
   execute BASE\LST\r_20_001.lst      ; запуск формирования отчета и печати его на принтере
 endif

 if (РАСЧ_РЕЖ_РТС (21,2,РЕЖИМ_РТС2,N2,L_РУЛЕТ_РТС2,ПНД_РТС2,
L_РТС2,4.7,5.9,-42.486,1548,-2859,-194.713,3234,-7505,T_РУЧ_РТС2,T_РТС2,
ПЛОТН20_РТС2,V_МАКС_РТС2,L_ТЕК_РТС2[VAG],M_ИЗМ_РТС2[VAG],
M_ИЗМ20_РТС2,ДОП_УТС_РТС2))
   sleep(5)
   execute BASE\LST\r_20_002.lst      ; запуск формирования отчета и печати его на принтере
 endif

 if (РАСЧ_РЕЖ_РТС (41,3,РЕЖИМ_РТС3,N3,L_РУЛЕТ_РТС3,ПНД_РТС3,
L_РТС3,4.7,5.9,-42.486,1548,-2859,-194.713,3234,-7505,T_РУЧ_РТС3,T_РТС3,
ПЛОТН20_РТС3,V_МАКС_РТС3,L_ТЕК_РТС3[VAG],M_ИЗМ_РТС3[VAG],
M_ИЗМ20_РТС3,ДОП_УТС_РТС3))
   sleep(5)
   execute BASE\LST\r_20_003.lst
 endif

; Расчет утечки за 20 минут и ведение очереди истории ------------------------
; AOUT[i] ... AUOT[19] - массив значений
; AOUT[i+20] - M_ИЗМ20_РТС изменение массы за 20 минут;
;-------------------------------------------------------------------------
func РАСЧ_УТЕЧ20 (i,РЕЖИМ_РТС,N,M_ТЕК_РТС,M_НАЧ_РТС)
 if eq(РЕЖИМ_РТС,0)
   if eq(N,1)                          ; режим хранение
     if ne(AOUT[i+19],0)               ; пока очередь не заполнится
       AOUT[i+20]=M_ТЕК_РТС-AOUT[i+19]  ; M_ИЗМ20_РТС
     else
       AOUT[i+20]=0                    ; утечки/перетока НЕТ!
     endif
     AOUT[i+19]=AOUT[i+18]           ; проталкиваем очередь
     AOUT[i+18]=AOUT[i+17]
     AOUT[i+17]=AOUT[i+16]
     AOUT[i+16]=AOUT[i+15]
     AOUT[i+15]=AOUT[i+14]
     AOUT[i+14]=AOUT[i+13]
     AOUT[i+13]=AOUT[i+12]
     AOUT[i+12]=AOUT[i+11]
     AOUT[i+11]=AOUT[i+10]
     AOUT[i+10]=AOUT[i+9]
     AOUT[i+9]=AOUT[i+8]
     AOUT[i+8]=AOUT[i+7]
     AOUT[i+7]=AOUT[i+6]
     AOUT[i+6]=AOUT[i+5]
     AOUT[i+5]=AOUT[i+4]
     AOUT[i+4]=AOUT[i+3]
     AOUT[i+3]=AOUT[i+2]
     AOUT[i+2]=AOUT[i+1]
     AOUT[i+1]=AOUT[i]
     AOUT[i]=M_ТЕК_РТС
   else
     if eq(N,0)                        ; режим инициализации
       InitOuts i, M_НАЧ_РТС, 20
     endif
   endif
 else                          ; РТС в резерве
   InitOuts i, 0, 20
endif
endfunc
```

```
; Расчет утечек РТС 56 ... РТС 58-------------------------------------------
 A=РАСЧ_УТЕЧ20 (1,РЕЖИМ_РТС56,N56,М_ТЕК_РТС56,М_НАЧ_РТС56)
 A=РАСЧ_УТЕЧ20 (22,РЕЖИМ_РТС57,N57,М_ТЕК_РТС57,М_НАЧ_РТС57)
 A=РАСЧ_УТЕЧ20 (43,РЕЖИМ_РТС58,N58,М_ТЕК_РТС58,М_НАЧ_РТС58)
```

12. Краткий справочник функций

Название	Параметры	Краткое описание	Возвращаемое значение
ABS	a	взятие модуля	
ACOS	a	арккосинус	
ASIN	a	арксинус	
ATAN	a	арктангенс	
BIT	dw,bit	выделить бит из двойного слова	
BITS	dw,bit,mask	выделить группу бит из двойного слова	
BXCHG	dw,byteseq	переставить байты в двойном слове	
COS	a	косинус	
CYCLESEC		получить период запуска алгоритма в секундах. Это величина указана в конфигурации алгоблока в тиках таймера.	
CURRENTSEC		возвращает прошедшее число секунд от запуска программы на исполнение	число секунд
DOST	a	определение признака достоверности выражения	
EQ	a,b	равно	
EXECSEC		получить измеренное время выполнения алгоритма в секундах.	
EXP	a	экспонента	
FALSE	a	значение выражения в скобках всегда будет недостоверно	
GE	a,b	больше или равно	
GETTICKS	prev_tick_cnt	получить число тиков таймера от момента запуска программы или предыдущее значение счетчика тиков таймера;	
GT	a,b	больше	
HOUR	a	где a = time() выделить часы.	
INITOUTS	i,v,c	инициализация массива OUT-ов значением v, начиная с i, c – штук.	
INT	a	взятие целой части	
LE	a,b	меньше или равно ,	
LOCK_ARCH	uso,dir	обозначить начало зоны работы с архивом УСО	
LN	a	натуральный логарифм	
LOG	a	десятичный логарифм	
LT	a,b	меньше	
MAKETIME	hour,min,sec, monthday,month,year	рассчитать время в секундах от 00:00:00 01/01/1970 года из пользовательских данных	
MAKETIMER	hour,min,sec, msec	рассчитать значение счетчика из пользовательских данных.	
MAX	a,b	максимум из двух чисел	
MIN	a,b	минимум из двух чисел	
MINUTE	a	выделить минуты, a = time()	от 0 до 59
MONTH	a	выделить номер месяца, a = time()	от 0 до 11
MONTHDAY	a	выделить день в месяце, a = time()	от 1 до 31
NE	a,b	не равно	
NMAX	a,b,…	максимум из рада чисел (у функции переменное число аргументов)	
NMIN	a,b,…	минимум из рада чисел (у функции переменное число аргументов)	
NOT	a	поразрядное инвертирование	
PIDREG	boi,S0_Ki,S1_Kp,S2_Kd,g,fu,y,y_diap,umin,umax,Tf,u_vmax,g_vmax,reg_mode,yf_size,yf_type,adapt_type,gf_ty	рассчитать значение сигнала управления	значение сигнала управления

Название	**Параметры**	**Краткое описание**	**Возвращаемое значение**
	pe,def_type,y_trust		
POW	a,b	возведение в степень	
RAND		получить псевдослучайное число	
REBOOT		мягкая перезагрузка (файловые операции закрываются),	
RESET	a	мягкая перезагрузка, вызываемая искусственным делением на 0	
RESTDIV	a,b	взятие остатка от деления	
SECOND	a	выделить секунды, a = time()	от 0 до 59
SET_UNCOND	sys,state	установить значение	
SET_WAIT	sys,state,time_out	установить значение	
SGN	a	взятие знака	
SIGN	a	взятие знака	
SIN	a	синус	
SQRT	a	квадратный корень	
STOP_SOFTDOG		перезагрузка, вызываемая срабатыванием программного сторожевого таймера потока алгоблока файловые операции закрываются, образуется файл coredump.txt	
TAN	a	тангенс	
TICKSIZE		получить интервал тиков таймера в секундах.	
TIME		получить текущее время в секундах от 00:00:00 01/01/1970 года;	
TIMERHOUR	a	получить число часов насчитанных счётчиком времени-a-значение счётчика времени	
TIMERMIN	a	получить число минут насчитанных счётчиком времени, a-значение счётчика времени	от 0 до 59
TIMERMSEC	a	получить число миллисекунд насчитанных счётчиком времени, a-значение счётчика времени	от 0 до 999.
TIMERSEC	a	получить число секунд насчитанных счётчиком времени, a-значение счётчика времени	от 0 до 59.
TRUE	a	значение выражения в скобках всегда будет достоверным	
UNLOCK_ARCH	uso, dir	обозначить конец зоны работы с архивом УСО	
WEEKDAY	a	получить номер дня в неделе, a = time()	0-Воскресенье, ..,6-Суббота
YEAR	a	выделить год, a = time()	4-х значный номер года (1970, 2010 и т.д.)
YEARDAY	a	получить номер дня в году, a = time()	от 0 до 365.