Встроенные процедуры в языке ST

Версия документа №1 от 14.11.06

1. Общие сведения о встроенных процедурах

Реализация языка ST в системе К748 предлагает пользователю набор готовых процедур (законченных программных блоков) предназначенных для решения различного рода часто встречающихся задач. Часть таких процедур реализована в компиляторе, а часть в микроядре реального времени процессорного модуля СР52.05. Все эти процедуры называются встроенными (встроенными в язык) в отличии от процедур которые могут быть самостоятельно написаны пользователем на некотором языке программирования (например ST).

С точки зрения языка ST все встроенные процедуры делятся на две группы: подпрограммы и функции. Отличия между ними в правилах обращения и способах обмена информацией. Функции, в языке ST, вызываются путем применения их в выражениях. Все параметры в функциях являются входными и в общем случае, могут быть произвольными выражениями. Список параметров, у некоторых функций, может быть пустым, т.е. не содержать ни одного параметра, но скобки должны присутствовать. Свое значение функция возвращает как бы через свое имя. Например: **i:= A*Sin(W*t+psi);** Здесь в выражении использовано обращение к функции синус.

Подпрограммы требуют для своего вызова применения отдельного оператора. Параметры подпрограмм могут быть как входными, так и выходными. С помощью последних подпрограммы возвращают вызывающей программе сформированные (вычисленные)значения. Все входные параметры, как и в случае функций, могут быть представлены произвольными выражениями. Выходные параметры могут быть только переменными. Так же существуют подпрограммы не имеющие параметров вообще (пустой список параметров), но скобки должны присутствовать.

Обратите внимание, что при обращении к процедурам, как встроенным так и к любым другим, необходимо строгое соблюдение правил вызова. То есть четкое соответствие между количеством, типом и последовательностью параметров, в противном случае компилятор выдаст сообщение об ошибке.

В настоящем документе приводится описание правил вызова и назначение всех встроенных подпрограмм и функций, соответствующий версии микроядра 2.42.

Все встроенные функции и подпрограммы здесь описаны в следующих форматах:

Имя_функции(Параметр1:Атрибуты1,Параметр2:Атрибуты2,..ПараметрN:АтрибутыN):Тип

Или:

Имя_подпрограммы(Параметр1:Атрибуты1,Параметр2:Атрибуты2,..ПараметрN:АтрибутыN)

Например:
Функция - **LOG(A:E_Real):Real**

Подпрограмма - **FillArrayW(A:A_Word,Code:E_Word);**

Здесь

LOG, FillArrayW – Соответственно имя функции и подпрограммы, те самые которое необходимо набирать в текстовом редакторе при написании программы;

A,Code – Условные идентификаторы параметров. Обратите внимание, обозначения A,Code выбраны исключительно условно для ссылки на них по тексту настоящего документа. Они не являются именами переменных вашего проекта. Вместо них вы должны будите вписать реальные параметры, в качестве которых, в общем случае, могут выступать константы, имена переменных и целые выражения. В каждом конкретном случае, для каждой конкретной подпрограммы вид параметра указывается описывается их атрибутами.

Real – Тип возвращаемого значения функции (представляется стандартными ST-типами). Вы не должны указывать его при вызове функции, он выполняет чисто описательную роль в настоящем документе, указывая вам какого типа будет операнд ассоциированный с именем данной функции в выражении.

E_Real,A_Word,E_Word – Атрибуты параметров, они определяют требования предъявляемые к параметрам и способ их интерпретации подпрограммой/функцией. Вы не должны указывать их при вызове подроограммы (функции), они выполняют чисто описательную роль в настоящем документе, указывая вам объекты какого типа разрешено использовать в качестве параметра. Подставляемый в вызов функции/подпрограммы параметр должен строго соответствовать требованиям указанных в описании атрибутов.

1.1. Условные обозначения атрибутов параметров.

Ниже приводится описание атрибутов, которые могут быть приставлены к параметрам подпрограмм и функций, описанных в настоящем документе.

Структура атрибута параметра имеет следующий вид:

<префикс>_<тип>;

где <*тип*>, это один из следующих стандартных ST – типов:

BOOL - он же бит (в словаре описывается как “Дискрет.”)
Byte (Байт без знака)
Word (16 бит без знака. В словаре описывается как “Аналог”)
Dword (двойное слово 32 бит без знака)
SINT (Short integer – байт со знаком)
INT (Integer – слово (16 бит) со знаком)
DINT (Double integer – двойное слово (32 бит) со знаком)
REAL (Вещественное 4х байтное)
SIGNED (Знаковый тип, любой из {Sint, Int, Dint})
UNSIGNED (Без знаковый тип, любой из {Byte, Word, Dword})
ALLINT (Целые, любой из {Byte, Word, Dword, Sint, Int, Dint})
ANY Любой тип

<*префикс*> может быть представлен одним из следующих символов с соответствующим значением:

:C константа (Constant);
:V переменная (Variable);

:A массив(array). За буквой "A" может следовать некоторое число, указывающее конкретную размерность массива который может быть применен в качестве параметра. Если конкретная размерность не указывается, значит в качестве параметра допускается массив произвольной длины (2…32000).

:E Выражение (Expression). Само сбой, разумеется, что там где допускается выражение там допускается и константа, и переменная, т.к. это частные случаи выражений.

Примеры атрибутов параметров:

:E_INT – выражение типа INT;
:A_Word – массив 16-ти битных слов;
:V_REAL – вещественная переменная.

---

2. Математические функции над целыми числами

2.2. Функция - **Abs(A:E_INT):Int**

Возвращает абсолютное значение выражения A типа INT
B:=Abs(A).
Особый случай : AbsI(-32768)=Abs(8000h)=+32767=7FFFh
Для справки : AbsI(-32767)=+32767

2.6. Функция - **LimitInt(Min:E_INT,In:E_INT,Max:E_INT):Int**

Возвращает ограниченное сверху и снизу значение входного выражения **In** типа **Int**.

```
LimitI = Min( Max(In,Min), Max);
```

Выходное значение функции находится в пределах Min<=Out<=Max.
Особый случай : если задано Max<Min, то LimitI =Max независимо от значения In.
Факт выполнения ограничения можно установить с помощью функции **LimitRes**.

2.7. Функция - **LimitW(Min:E_WORD,In:E_WORD,Max:E_WORD):Word**

Возвращает ограниченное сверху и снизу значение входного выражения **In** типа **Word**.

```
LimitW = Min( Max(In,Min), Max);
```

Выходное значение функции находится в пределах Min<=Out<=Max.
Особый случай : если задано Max<Min, то LimitW =Max независимо от значения In.
Факт выполнения ограничения можно установить с помощью функции **LimitRes**.

2.8. Функция - **LimitRes():Word**

Возвращает в признаки выполнения ограничения в последней LimitX в следующем формате :

```
15 ... 3     2    1    0
 0 ... 0   Limit Max  Min
```

Бит Min=1 если выполнено ограничение "снизу", т.е. In<Min и In

заменено на Min
Бит Max=1 если выполнено ограничение "сверху",т.е. In>Max и In заменено на Max
Бит Limit=1 если выполнено ограничение "снизу" или "сверху"

Примечание. Если в LimitX задано Min>Max, то настоящий регистр равен 0007h.

2.9. Функция – **ShL(A:E_ALLINT, B:E_ALLINT):ALLINT**

Логический сдвиг влево. Возвращает значение выражения “A”, сдвинутое влево (в сторону старших разрядов) на количество бит определяемых выражением “B”. Освобождающиеся биты заполняются нулем. Само “A” остается неизменным. Тип возвращаемого значения равен типу выражения “A”.

2.10. Функция – **ShR(A:E_UNSIGNED, B:E_ALLINT):UNSIGNED**

Логический сдвиг вправо. Возвращает значение выражения “A”, сдвинутое вправо (в сторону младших разрядов) на количество бит определяемых выражением “B”. Освобождающиеся биты заполняются нулем. Само “A” остается неизменным. Тип возвращаемого значения равен типу выражения “A”.

2.11. Функция – **ShRA(A:E_SIGNED, B:E_ALLINT):SIGNED**

Арифметический сдвиг вправо. Возвращает значение выражения “A”, сдвинутое вправо (в сторону младших разрядов) на количество бит определяемых выражением “B”. Освобождающиеся биты заполняются знаком числа. Само “A” остается неизменным. Тип возвращаемого значения равен типу выражения “A”.

3. Математические процедуры над вещественными числами

3.0. Общие сведения о представлении вещественных чисел в ср59.05

Вещественные числа представляются согласно формату "Короткое вещественное
длиной 32b" стандарта IEEE-754(854) (Single Float).

Количество значащих цифр представления 7..8.
Диапазон представления 1.1754944351e-38 ... 3.402823466e+38.

Операции с вещественными числами проводятся с помощью математического
сопроцессора FPU (Floating Point Unit - СОПР) соответствующего стандартам
IEEE-754 и 854.

Младший бит результата арифметических операций округляется к ближайшему
числу.

При выполнении вычислений над вещественными числами результат может

оказаться непредставимым в разрядной сетке (например деление на 0 или корень квадратный из отрицательного числа),- при этом FPU, в качестве результата, формирует так называемое особое число, которое, в дальнейшем,
обрабатывается без останова вычислений, но все последующие результаты, где используется это число, будут также носить признак особого числа.

Если входные данные расчетного алгоритма таковы, что могут проявиться эти особые случаи, то пользователь должен самостоятельно, в выбранных им точках, контролировать результаты расчетов на предмет особых случаев и предпринимать соответствующие меры, в случае их выявления.

Проконтролировать результат можно с помощью СФ StatFPU, которую следует вызывать непосредственно после выполнения СФ над вещественным числом.

Определить тип вещественного числа (особое оно или нет), - в любой момент, также можно и с помощью СФ FXAM.

3.1. Функция - **StatFPU():Word**

Возвращает значение регистра состояния операций с вещественными числами, при этом этот регистр сбрасывается.
Кроме того, этот регистр автоматически сбрасывается :
-- в начале каждого скана ПЛК
-- в начале выполнения каждой СФ над вещественным числом.

3.1.1. Формат регистра состояния операций с вещественными числами :

Бит	Имя	Описание
0	Invalid	1 - произошла недопустимая операция -- корень квадратный или логарифм от отрицательного числа -- тригонометрическая операция над бесконечностью -- выполнена запрещенная операция над особыми числами (например сложение положительной и отрицательной бесконечностей) -- результат IFIX вне диапазона представления 16-битного целого со знаком
1	Over	1 - деление на ноль или переполнение; далее в вычисления будет передано особое число - бесконечность
2	Under	1 - антипереполнение : результат настолько мал, что находится между нулем и минимальным по модулю числом, представимым в СОПР; далее в вычисления будет передан ноль
3	Precision	1 - потеря точности; например при сложении 1e9 и 1 результат будет равен 1e9 и бит Precision=1
4	Trigon	1 - аргумент тригонометрической функции вне допустимых пределов -2**63 .... +2**63 радиан; после нетригонометрических операций значение этого бита м.б. любым.

```
   5..7   резерв

        Результаты сравнения 2х вещественных чисел :
   8    NC     1  A и B несравнимы, т.е. или A или B являются особым
числом;
                  (Если бит NC=1, то и бит Invalid=1)
   9    EQ     1  A=B
  10    NE     1  A<>B
  11    LT     1  A<B
  12    LE     1  A<=B
  13    GT     1  A>B
  14    GE     1  A>=B
  15   резерв

  Примечания. 1) Биты 8..14 заносятся только при выполнении СФ сравнения
                 вещественных чисел в СФ CMPR,FTST; функция StatFPU
                 возвращает в этих битах 0.
              2) Если бит 9..14 равен "1", то соответствующее
соотношение
                 выполнено. Одновременно устанавливаются биты ВСЕХ
                 выполненных соотношений.
                 Например если A=5 и B=6, то NE=1, LT=1, LE=1 ;
                          если A=5 и B=5, то EQ=1, LE=1, GE=1.
-------------------------------------------------------------------
Группа функций преобразования типов

   Byte_TO_ Real(A:E_Byte):Real;
   Word_TO_Real(A:E_Word):Real;
   Dword_TO_Real(A:E_Dword):Real;
   Sint_TO_Real(A:E_Sint):Real;
   Int_TO_Real(A:E_Int):Real;
   Dint_TO_Real(A:E_Dint):Real;
   Real_TO_Word(A:E_Real):Word;*
   Real_TO_Int(A:E_Real):Int;*
   Real_TO_Dint(A:E_Real):Dint;
```

Настоящие функции преобразуют соответствующий тип параметра к соответствующему типу возвращаемого значения. При преобразовании вещественного выражения в целые типы, вещественное округляется до ближайшего целого :

```
    +12345.4 -> +12345
    +12345.5 -> +12346
    -12345.4 -> -12345
    -12345.5 -> -12346
    -0.0999  -> 0
   Если результат оказывается в диапазоне представления возвращаемого
типа, то бит StatFPU.Invalid=0; если результат вне диапазона
   представления, то бит StatFPU.Invalid=1.
     А сами возвращаемые значения будут следующими:
     Функция Real_TO_Word : - 0 (Ноль)
     Функция Real_TO_Int : - +32767 если A>0
                         или -32768 если A<0

     Функция Real_TO_Dint : +2147483647 если A>0
                          или -2147483648 если A<0
```

3.6. Функция - **FXAM(A:E_Real):Word**

Определение типа (статуса) вещественного выражения A.
Возвращает значение регистра типа вещественного числа.

3.6.1 Формат регистра типа вещественного числа

Бит	Имя	Описание
0	Normal	1 - нормальное конечное число или 0 или денормализованное число (число с нулевой характеристикой - весьма малое число)
1	Infinity	1 - бесконечность (положительная или отрицательная)
2	Special	1 - не-число (иные,особые числа)
3	NoSupport	1 - не поддерживаемое число (число вне формата СОПР)

3.16. Функция - **SQRT(A:E_REAL):REAL**
Возвращает квадратный корень из параметра A.

3.17. Функция - **AbsR(A:E_REAL):REAL**
Возвращает абсолютное значение вещественного A.

3.19. Функция - **LN(A:E_REAL):REAL**
Возвращает натуральный логарифм A : B=Ln(A)
Особые случаи :
Если A<0 , то бит StatFPU.Invalid=1.
Если A=0 , то бит StatFPU.Over=1.

3.20. Функция - **LOG(A:E_REAL):REAL**
Возвращает десятичный логарифм A (по основанию 10): B=Lg(A)
Особые случаи :
Если A<0 , то бит StatFPU.Invalid=1.
Если A=0 , то бит StatFPU.Over=1.

3.21. Функция - **EXP(A:E_REAL):REAL**
Возвращает експоненту выражения A : B=Exp(A), т.е. натуральное основание E в степени A.
Особые случаи :
- если результат менее нижней границы диапазона представления вещественных чисел (антипереполнение), то бит StatFPU.Under=1, что в инженерных приложениях игнорируется.
- если результат более верхней границы диапазона представления вещественных чисел (переполнение), то бит StatFPU.Over=1.

3.22. Функция - **SIN(A:E_REAL):REAL**
Возвращает синус выражения A : B=Sin(A)
A д.б. задано в радианах
Если A вне интервала -2**63 ... +2**63 (приблизительно : -9.22e18 ... +9.22e18), то СФ не выполняется :
- B=A
- бит StatFPU.Trigon=1.

3.23. Функция - **COS(A:E_REAL):REAL**
Возвращает косинус выражения A : B=Cos(A)
A д.б. задано в радианах

Если A вне интервала -2**63 ... +2**63 (приблизительно :
-9.22e18 ... +9.22e18), то СФ не выполняется :
- B=A
- бит StatFPU.Trigon=1.

3.24. Функция - **TAN(A:E_REAL):REAL**

Возвращает тангенс выражения A : B=Tg(A)
A д.б. задано в радианах
Особые случаи :
- если A вне интервала -2**63 ... +2**63 (приблизительно :
  -9.22e18 ... +9.22e18), то бит StatFPU.Trigon=1
- если A=(+-Pi/2+N*Pi), где N-целое число, то бит StatFPU.Invalid=1

3.25. Функция - **ASIN(A:E_REAL):REAL**

Возвращает главный арксинус A : B=ArcSin(A).
A м.б. в интервале : -1.0 ... +1.0.
B=-Pi/2 ... +Pi/2, радиан.
Если A вне интервала -1.0 ... +1.0 , то СФ не выполняется :
- B не определено
- бит StatFPU.Invalid=1 и StatFPU.Trigon=1

3.26. Функция - **ACOS(A:E_REAL):REAL**

Возвращает главный арккосинус A : B=ArcCos(A).
A м.б. в интервале : -1.0 ... +1.0.
B= 0 ... +Pi, радиан.
Если A вне интервала -1.0 ... +1.0 , то СФ не выполняется :
- B не определено
- бит StatFPU.Invalid=1 и StatFPU.Trigon=1

3.27. Функция - **ATAN(A:E_REAL):REAL**

Возвращает главный арктангенс A : B=ArcTg(A).
A м.б. в интервале : -oo ... +oo.
B=-Pi/2 ... +Pi/2, радиан.

3.28 Функция - **EXPT(A:E_REAL, Y:E_REAL):REAL**

Возвращает произвольную степень выражения A : B=A**Y

Определение функции A**Y:

A	Y	B
0	<0	не определена(ошибка)
0	0	1
0	>0	0
любое(A<>0)	целое	B=A**Y
любое(A<0)	нецелое	не определена(ошибка)
любое(A>0)	нецелое	B=A**Y

При ошибке устанавливается бит StatFPU.Invalid=1

3.29 Подпрограмма - **IntFrac(A:E_REAL, B:V_REAL, C:V_REAL)**

Разлагает вещественное A на целую B и дробную C части.
B и C имеют знак числа A и также вещественные.

B+C=A.

3.30 Функция - **Sqr(A:E_REAL):REAL**

вычисляет квадрат A : B=A*A

выполняется быстрее чем EXPT

3.31 Функция - **Round(A:E_REAL):REAL**

Округляет A к ближайшему целому и возвращает результат.

Примеры :

- Round(+1.234)=+1.0
- Round(+1.500)=+2.0
- Round(-1.234)=-1.0
- Round(-1.500)=-2.0
- Round(+0.4) = 0
- Round(-0.4) = 0

3.32 **LimitR(Min:E_REAL, In:E_REAL, Max:E_REAL):REAL**

Возвращает ограниченное сверху и снизу значение входного выражения **In** типа **Real**.

LimitR = Min(Max(In,Min), Max);

Выходное значение функции находится в пределах Min<=Out<=Max.

Особый случай : если задано Max<Min, то LimitR =Max независимо от значения In.

Факт выполнения ограничения можно установить с помощью функции **LimitRes**.

---

4. Диагностические процедуры

4.1. Подпрограмма - **DgnPLC(A:V_Word)**

Получение обших данных о состоянии ПЛК :
пересылает в переменную A основной регистр (слово) диагностики ПЛК
после чего сбрасывает бит Net в этом регистре.

Структура основного слова диагностики ПЛК (DgnPLC)

"1" в бите означает наличие ненормы(события)

Бит	Имя	Описание
0	CMOS	ОРИ или отказ CMOS05, уточнить причину ненормы РП можно СФ DgnPLCRg(DgnCMOS05).
1	FileRP	Ошибка чтения файла с РП - при этом загружается пустой проект; Кроме дисковых ошибок обмена также считается ошибкой длина файла РП менее минимальной или более максимальной; CRC РП будет проверена при запуске РП. При этой ошибке блокируется автозапуск РП.
2	RP	Ненорма РП, уточнить причину ненормы РП можно СФ DgnPLCRg (DgnRP1,DgnRP2)

3	RTC	Отказ часов реального времени (RTC) - уточнить причину ненормы можно СФ DgnPLCRg(DgnCMOS05).
4	LowPC104	Высокая скорость обмена через PC104 не устанавливается : модуль ПЛК неработоспособен
5		резерв
6	PVC	ПВЦ - превышение максимального времени скана (цикла). Значение этого времени устанавливается пользователем в K748. Получить фактическое значение времени скана при ПВЦ можно СФ DgnPLCRg(DgnPVC)
7		резерв
8	STP	Останов РП от команды СТП (применяется пользователем)
9	OVV	ошибка ввода-вывода (ОВВ) в главном каркасе - уточнить место каркаса можно СФ Units(0)
10	UART	Ненорма самопроверки UART C750 для расширителей или ненорма передатчика в процессе РП - уточнить причину ненормы можно СФ DgnPLCRg(Dgn750Rash).
11	NoRashs	Нет связи с одним из расширителей (для уточнения см. СФ DgnExpI)
12	OVVRashs	Ошибки ввода/вывода в каркасе одного из расширителей - уточнить место можно СФ Units(n)
13	EthNet	Ошибки обмена ModBus TCP/IP/EtherNet (для уточнения см. СФ DgnEthNet)
14	Net	Ошибка сетевой диагностики модулей 52.05/52.07 (см. слова сетевой диагностики модулей 52.05/52.07 - СФ DgnNet или регистры диагностики СФ ReadNet,WriteNet,SnglWNet)
15		резерв

Если в конфигурации проекта задан "СТОП при отказе ОРИ", то ПЛК всегда переходит в СТОП при ненорме в бите CMOS(0).

При получении ненорм в битах PVC(6),STP(8) ПЛК всегда переходит в СТОП.

При получении ненормы в бите RP(2) ПЛК всегда переходит в СТОП, кроме случаев указанных в описаниях DgnRP1,DgnRP2 - когда ненорма просто помечается в соответствующем регистре DgnRPi и бите DgnPLC.RP.

Если в конфигурации проекта задан хотя бы один расширитель, то ПЛК всегда переходит в СТОП при ненорме в бите UART(10).

Если в конфигурации проекта задан СТОП при ОВВ хотя бы в одном модуле главного каркаса, то ПЛК всегда переходит в СТОП при ненорме в бите OVV(9).

Если в конфигурации проекта задан СТОП по ОВВ хотя бы в одном модуле расширителя и этот расширитель включен в конфигурацию, то ПЛК всегда переходит в СТОП при ненорме в битах NoRashs(11),OVVRashs(12).

При ненорме в бите Net(14) ПЛК никогда не переходит в СТОП

4.2. Подпрограмма - **DgnPLCRg(A:A10_Word)**

Записывает в массив A все регистры(слова) диагностики ПЛК
Массив занимает 10 слов

Структура массива :

Адрес слова	Имя	Назначение
A	DgnPLC	Основное слово диагностики ПЛК (отдельно его так же можно получить функцией DgnPLC)
A+1	DgnRP1	Отказы РП, слово 1
A+2	DgnRP2	Отказы РП, слово 2
A+3	DgnAdrRP	Адрес отказа РП (смещение от 1го байта РП) - Low часть
A+4		Адрес отказа РП (смещение от 1го байта РП) - High часть
A+5	DgnPVC	Время скана при ПВЦ, цмр=1мсек
A+6	DgnCMOS05	Ненормы энергонезависимой батарейной памяти ПЛК (CMOS05) Ненормы часов реального времени (RTC)
A+7	Dgn750Rash	Ненормы приемопередатчика сети расширителей
A+8		резерв
A+9		резерв

Структура основного слова диагностики ПЛК приведена в описании СФ DgnPLC.

Ненормы, которые могут возникнуть при выполнении РП, делятся на два класса :
- фатальные : приводят к немедленному прекращению выполнения РП
- этапа выполнения РП : помечаются в бите DgnPLC.RP и в одном из битов DgnRPi; такие ненормы фиксируются накапливающим способом до перезапуска РП или до выполнения спецфункции ResetDgnRP; они отмечены ниже, в описаниях регистров DgnRPi, знаком "!".

При возникновении ненормы РП :
- устанавливается в "1" бит основного слова диагностики ПЛК DgnPLC.RP
- устанавливается в "1" соответствующий бит в DgnRP1 или DgnRP2
- в Адрес отказа РП DgnAdrRP заносится 32х битное смещение начала команды, при выполнении которой произошла ненорма, от 1го байта РП (паспорт не учитывается)
- выполнение РП прекращается или продолжается несмотря на наличие ненормы (только для ненорм "!" этапа выполнения РП).

Бит DgnPLC.RP, регистры DgnRPi устанавливаются в норму ("0") при каждом запуске РП или могут быть отдельно обнулены с помощью спецфункции ResetDgnRP.

!

Структура слова диагностики ПЛК "Отказы РП, слово 1" (DgnRP1):

Бит	Имя	Описание
0	CRC	Ненорма CRC РП
		Ошибки этапа предварительного скана
1	OutRP	Не найдено окончание РП (команда КМП)
2	BigSgm	Номер сегмента более максимального
3	AlrSgm	Сегмент уже определен
4	ErrorKOP	Недопустимый код команды
5	BigBlock	Номер блока более максимального
6	AlrBlock	Блок уже определен
7	AlrProc	ПП уже определена
8	ErrorSF	СФ не реализована
		Ошибки этапа выполнения РП
9	OutRP	Выполнение команды вне пределов РП
10	ErrorKOP	Недопустимый код команды
11	ErrorSF	СФ не реализована
12	FunStackG	стек вызова подпрограмм исчерпан
13	FunStackL	попытка возврата из подпрограммы при пустом стеке
14	StackErr	при выходе из РП не выравнен стек, указатель стека при входе и при выходе имеет разные значения
15	OutMemGP	при выполнении РП произошло обращение к данным вне памяти ОЗУ ПЛК (исключение GP 0Dh)

Структура слова диагностики ПЛК "Отказы РП, слово 2" (DgnRP2):

Бит	Имя	Описание
0	CMR	неверный код ЦМР в команде таймера
1		резерв
2!	SZone	нарушена S-зона данных усреднения
3!	SZonePrm	ошибка во входных параметрах для подпрограммы усреднения AvgMVxx
4!	RTCData	ошибка в данных одной из спецфункций RTC(часов реального времени)
5	NoProc	вызов несуществующей подпрограммы
6!	OutArray	обращение вне пределов массива
7	NoEqKrnVer	не совпали текущая версия ядра ПЛК и версия ядра,на которой загружался проект : ВЫПОЛНИТЬ ЗАГРУЗКУ ПРОЕКТА
8..15		резерв

Структура слова диагностики ПЛК "Ненормы энергонезависимой батарейной памяти ПЛК (DgnCMOS05)":

Бит	Имя	Описание
		Биты диагностики ненорм CMOS05
0	ORI	ОРИ - отказ резервного источника питания (батареи)
1	WrRd	Записанный в CMOS05 код не совпал со считанным
2	KS	Ненорма контрольной суммы CMOS05 при восстановлении данных из CMOS05
3..7		резерв
		Биты диагностики ненорм часов реального времени RTC
8	AB	Ненорма Аккумуляторной Батареи питания RTC
9	TimeOut	Тайм-аут при обращении к порту RTC
10	Int	Нет прерывания от RTC за время 3.5 сек
11..15		резерв

Структура слова диагностики ПЛК "Ненормы приемопередатчика сети расширителей" (Dgn750Rash) :

Бит	Имя	Описание
0	SCR	Ненорма проверки записи/чтения в SCR C750 (C750 отсутствует или неисправен)
1	FIFO64	Ненорма проверки установки режима FIFO 64b (установлен C?50, который не поддерживает режим FIFO 64b)
2	RTS	Ненорма проверки RTS (неисправен 750)
3	Trans	Нет пересылки байта (неисправен 750)
4	Fosc	Ненорма проверки длительности пересылки байта (неисправен 750 или Fosc750<16МГц)
5	ErrCode	Контрольный байт принят с искажениями (неисправен 750)
6,7		резерв
8	ErrTS2	Нет срабатывания системного IBM-таймера Sys2 при отсчете задержки
9	ErrTxd	Передатчик C750 за контрольное время не передал байт
10..15		резерв

Примечание:
- в битах 0..7 расположены признаки, выявляемые при самопроверке микросхемы TL16C750 (UART сети RS485 расширителей), когда запускается РП
- в битах 8..15 расположены признаки, выявляемые в процессе работы РП
- наличие отказов на самопроверке блокирует РП, если есть расширители
- отказы в процессе работы РП просто фиксируются, т.к. они проявятся через контроль передачи фреймов.

4.3. Подпрограмма – **ResetDgnRP()**

Сброс ненорм этапа выполнения РП :
- Сбрасывает DgnPLC.RP в 0
- Сбрасывает биты DgnRP1, DgnRP2 ненорм этапа выполнения РП (не

приводящие к стопу РП) в 0
- Сбрасывает DgnAdrRP в 0

4.4. **Units(Karkas : W; Units : WV)**

Получение данных о расположении отказавшего модуля (виртуального модуля для блочных К120) :
пересылает в Units слово отказов модулей в главном каркасе(ветке для К120) или расширителе.

Karkas = 0 для главного каркаса(ветки)
Karkas = 1..7 для каркасов(веток) расширителей

"1" в бите N слова Units означает ошибку ввода вывода (ОВВ) в модуле на месте N (0...15)

4.5. **Units2(Vetka : W; Units : WV) [только для блочных К120]**

Получение данных о расположении отказавшего виртуального модуля :
пересылает в Units слово отказов вирт.модулей в главной ветке или в ветках расширения.

Vetka = 0 для главной ветки
Vetka = 1..4 для веток рсширения

"1" в бите N слова Units означает ошибку ввода вывода (ОВВ) в виртуальном модуле на месте N+16 (16...31)

4.6. Подпрограмма – **DgnExp(A:V_Word)**

Пересылает в переменную А регистр наличия ошибок связи по интерфейсу RS485 всех расширителей

Структура регистра :

Бит		
0		Резерв
1	"1"	Имеются ошибки связи с расширителем 1
2	"1"	Имеются ошибки связи с расширителем 2
...		
7	"1"	Имеются ошибки связи с расширителем 7
8		Резерв
...		
15		Резерв

Если бит взведен, то произошла одна из ненорм :
- За контрольное время не начал поступать ответ из РАСШ при чтении из него фрейма-данных
- Не совпала CRC при приеме фрейма (после 3х повторов)
- Данные "Из РАСШ" не соответствуют данным "В РАСШ" (после 3х повторов)
- Не совпала CRC при приеме фрейма в РАСШ (после 3х повторов)

Для уточнения причины установки бита - см. DgnExpI.

4.7. Подпрограмма – **DgnExpI(Karkas:E_Word, DgnExpI:V_Word)**

Пересылает в переменную DgnExpI регистр причины ошибки связи по интерфейсу RS485 с заданным расширителем Karkas (1..7)
Если задан номер каркаса 0 или более 7, то возвращается 0.

Структура регистра причины ошибки связи :

Бит 0 "1" После ответа из 2b не принят 3й BI -байт
1 "1" За контрольное время не начал поступать ответ из РАСШ при чтении из него фрейма-данных
2 "1" Ошибка данных в принятом из RXD байте (BI,FE,PE,OE)
3 "1" Не совпала CRC при приеме фрейма (после 3х повторов)
4 "1" Перерыв в поступлении байт фрейма более TSilence
5 "1" Поступил "чужой" адрес
6 "1" Поступил неверный код функции
7 Резерв
8 "1" Данные "Из РАСШ" не соответствуют данным "В РАСШ" (после 3х повторов)
9 "1" Не совпала CRC при приеме фрейма в РАСШ (после 3х повторов)

4.8. Подпрограмма – **PuskPLC(NumberPusk:WV)**

Возвращает текущий номер запуска ядра ПЛК 59.05 в переменной NumberPusk
Номер хранится в энергонезависимой памяти.
Внешний наблюдатель, контролируя этот номер может фиксировать перезапуски ПЛК (ручные, от Watch Dog Timer'a);

Структура NumberPusk :
Биты 0..7 Номер запуска
Биты 8..15 Резерв

4.9. Подпрограмма - **DgnHSBRg (A:A6_Word)**

Записывает в массив A длиной 6 слов след. слова диагностики системы горячего резервирования (HSB):

Структура массива :

Адрес слова	Имя	Назначение
A+0	DgnHSB	Обобщенные отказы системы горячего резервирования (HSB).
A+1	LPT_Status	Отказы LPT порта.
A+2	DgnSwitchHSB	Причины переключений “Ведущий -> Резервный”.
A+3	DgnUSBSys	Текущие ненормы USB системного уровня.
A+4	DgnUSBIni	Ошибки этапа инициализации USB.
A+5	DgnUSBTmp	Ошибки в процессе работы USB.

Структура слова “обобщенные отказы системы горячего резервирования(DgnHSB)”:

Бит	Описание
0	Ненорма LPT.
1	Ненорма USB или обмена по USB.
2	Использован контур ПИД с номером превышающим максимальное число контуров FactNumKonturPID указанное в конфигурации.
3	В конфигурации РП отключен WDT, HSB-Функционирование при отключенном WDT НЕ ПРОИЗВОДИТСЯ !!!
4	Недопустимое взаимное HSB-состояние ПЛК.
5	Недостоверный HSB-статус ПЛК, для ПЛК навязан статус Defect0.
6	Версии системных программ ПЛК (ядер) не идентичны.
7	Рабочие программы ПЛК не идентичны.
8	Ошибка при копировании РП в другой ПЛК.
9	Тумблер в положении "ОТЛ".
10..14	Резерв
15	В данном ПЛК произошло переключение ПЛК: - Этот ПЛК прекратил управлять объектом - Другой ПЛК стал управлять объектом.

Структура слова “Отказы LPT порта(LPT_Status)”:

Бит	Описание
0	LPT не обнаружен (Базовый адрес LPT не равен 0378h или 0278h)
1	В BIOS Setup для LPT не задан режим ECP.
2	Для LPT не удалось установить режим BiDirect из режима ECP.
3..15	Резерв.

Структура слова “ Причины переключений “Ведущий -> Резервный” (DgnSwitchHSB)”:

В младшем байте, настоящего слова, содержится код причины переключения. Старший байт слова содержит счетчик переключений.

Коды переключений и их значения приведены ниже:

Код	Описание
0	Переключения не было.
1	Отказ линии связи с одним из каркасов расширения.
2	Отказ модуля в главном каркасе : получена ОВВ для модуля, у которого заказано "Переключение на Резервный ПЛК при ОВВ".
3	Отказ (тайм-аут) связного модуля при обмене по внутрикаркасному интерфейсу.

4	Превышение Времени Цикла (скана).
5	Ненорма самопроверки UART TL16C750 для расширителей.
6	Ненорма записи данных в CMOS,если установлен "СТОП ПЛК при ОРИ"
7	При выполнении РП возникло GP-исключение.
8	Ошибка чтения файла РП с диска при автозагрузке РП.
20	Ненорма CRC РП.
21	Не найдено окончание РП (команда КМП).
22	Номер сегмента более максимального.
23	Сегмент уже определен.
24	Недопустимый код команды.
25	Номер блока более максимального.
26	Блок уже определен.
27	ПП уже определена.
28	Встроенная (специальная) Функция не реализована.
29	Выполнение команды вне пределов РП.
30	Недопустимый код команды.
31	СФ не реализована.
35	При выполнении РП произошло обращение. к данным вне памяти ОЗУ ПЛК- исключение GP.
45	Вызов несуществующей подпрограммы.
46	Обращение вне пределов массива.

Структура слова " Текущие ненормы USB системного уровня (DgnUSBSys)":

Бит	Описание
0	Отсутствуют оба HSB кабеля.
1	В последнем сеансе обмена не получен ответ партнера.
2	Отсутствует Левый HSB кабель
3	Отсутствует Правый HSB кабель
4	Ошибка переинициализации кабеля 0
5	Ошибка переинициализации кабеля 1
6	отсутствует канал 0
7	отсутствует канал 1
8..15	Резерв

Структура слова " Ошибки этапа инициализации USB (DgnUSBIni)":

Бит	Описание
0	В BIOS нет PCI функций.
1	Не найден USB-контроллер SiS.
2	USB контроллер не OHCI.
3	Ненорма Irq Line отведенного BIOS\Plag&Play для USB-контроллера - не удалось считать PCI.Config.IrqLine для USB - Irq вне диапазона [10,11,12] - Irq уже занят другим устройством. Эта ненорма возникает при установке в BIOS раздел "PCI/Plug and Play Setup" для Irq10,11,12 не "PCI/Pnp", а "ISA/EISA"
4	USB контроллер не OHCI (hcRevision не равен 0000 0110h)
5	Не получено подтверждение о сбросе HC или

	HC не перешел в UsbSuspend
6	HC не перешел в UsbOperational
7	За тайм-аут, не изменился hcFmRemaining (нет подтверждения наличия сетки в линию USB)
8	Превышение тока потребления устройством на порту 0
9	Превышение тока потребления устройством на порту 1
10	Тайм-аут при программной инициализации активного кабеля на порту 0
11	Тайм-аут при программной инициализации активного кабеля на порту 1
12	Не включился порт 0 USB
13	Не включился порт 1 USB
14	К порту 0 не подстыкован кабель ПЛК-ПЛК
15	К порту 1 не подстыкован кабель ПЛК-ПЛК

Структура слова " Ошибки в процессе работы USB (DgnUSBTmp)":

Бит	Описание
0	На стороне Mastera для передачи задано число байт превышающее размер буфера.
1	Halt в Host USB-контроллере (отстыкован кабель от другого ПЛК или ошибка в дескрипторе настоящего ПЛК).
2	Превышение тока потребления кабелем ПЛК-ПЛК.
3	Нет ответа на запрос в кабель ПЛК-ПЛК.
4	Данные не переданы за тайм-аут.
5	Ответчик вернул NAK("Занято")
6	нарушение или рассинхронизация обмена через USB-кабель ("В принятом пакете обнаружена ошибка", "От партнера получена неверная команда", "Неверная квитанция").
7	На стороне Slave-а получено число байт отличающееся от числа, заданного Master-ом.
8	К порту 0 не подстыкован кабель ПЛК-ПЛК (нет соединения с USB-Device).
9	К порту 1 не подстыкован кабель ПЛК-ПЛК (нет соединения с USB-Device).
10	Отключен порт 0 USB.
11	Отключен порт 1 USB.
12	Внутренняя ошибка USB-мо.
13	Резерв
14	Данный бит устанавливается ВНЕ мо USB - в pHsbLink Если=1, то номер транзакции посланный в Slave не равен номеру транзакции возвращенному из Slave.
15	Резерв.

5. Сетевые процедуры ModBus

5.0. Регистр диагностики сетевых функций ПЛК ReadNet, WriteNet для модулей 52.05/52.07

Регистр диагностики определяет завершена ли запущенная в сети ModBus операция и с какими результатами, когда ПЛК - активная станция.

Для обменов, когда ПЛК пассивная станция, ненормы фиксируются другим способом (см. ActNet, DgnNet).

Расположение этого регистра указывает пользователь при запуске операции.

При запуске операции этот регистр обнуляется.

Выполнив заданную операцию ПЛК должен сформировать в этом регистре данные о результатах выполнения :

```
                15 14 ... 8 7 ..3 2 1 0
                 k  k ... k x   B E T N
```

N - 1 получен нормальный ответ
T - 1 нет связи (тайм-аут ПЛК)
E - 1 получен особый ответ (сообщение об ошибке)
B - 1 отказ в запуске СФ :
-- на заказанном месте каркаса нет 52.05/52.07
-- номер канала не равен 1 или 2
-- адрес в сети ModBus более 31
-- число регистров более 123
-- выходные данные сетевой СФ вне ТД
-- число заказанных регистров ModBus более длины массива буфера данных
-- предыдущая операция не завершена (канал еще занят)
-- нет готовности 52.05/52.07 к приему за 6ms при запуске операции

k - байт кода особого ответа (сообщения об ошибке)
01 ILLEGAL FUNCTION
02 ILLEGAL DATA ADDRESS
03 ILLEGAL DATA VALUE (затребовано более 123 регисторв)
04 SLAVE DEVICE FAILURE (ответный ПЛК занят - в 52.05/52.07 тайм-аут 170ms)

5.1. Подпрограмма –

ReadNet(Mesto:E_Word, Kanal:E_Word, Abonent:E_Word,
BegRg:E_Word, NumRg:E_Word,
Cosv:C_Word, Buf:A_Word,
Dgn:V_Word, TimeOut:E_Word)

Чтение через заданный канал связного модуля 52.05/52.07 данных пассивного абонента.

Подпрограмма :
- запускает процесс чтения (ModBus Func03)
- контролирует окончание чтения или истечение тайм-аута на чтение
- заносит считанные данные в буфер ТД
- формирует в отведенном пользователем регистре диагностики линии связи результат операции

Параметры :
Mesto - номера места установки модуля в корзине ПЛК (0..15)

Kanal - номер канала в 52.05/52.07 (1 или 2)
Abonent - номер абонента в сети ModBus (1..31), подключенной к заданному каналу модуля 52.05/52.07
BegRg - начальный номер Z-регистра в абоненте (0..65535), начиная с которого считывать данные
NumReg - число читаемых в абоненте Z-регистров (1..123)
Cosv - признак косвеннной адресации буфера приема данных Должен быть 0 т.е допускается только прямая адресация

Buf - Массив - буфер приема данных из пассивного абонента.

TimeOut - Тайм-аут,мсек (50...65535)

Dgn - регистр диагностики линии связи, при запуске операции сюда записывается 0, по окончании операции сюда заносится ее результат (см. "Регистр диагностики сетевых функций ПЛК ReadNet, WriteNet")

5.2. Подпрограмма –
WriteNet (Mesto:E_Word, Kanal:E_Word, Abonent:E_Word, BegRg:E_Word, NumRg:E_Word, Cosv:C_Word, Buf:A_Word, Dgn:V_Word, TimeOut:E_Word)

Запись через заданный канал связного модуля 52.05/52.07 данных в пассивный абонент.

Подпрограмма :
- запускает процесс записи (ModBus Func16)
- контролирует окончание записи или истечение тайм-аута на запись
- формирует в отведенном пользователем регистре диагностики линии связи результат операции

Параметры :
Mesto - номера места установки модуля в корзине ПЛК (0..15)
Kanal - номер канала в 52.05/52.07 (1 или 2)
Abonent - номер абонента в сети ModBus (1..31), подключенной к заданному каналу модуля 52.05/52.07
BegRg - начальный номер Z-регистра в абоненте (0..65535), начиная с которого записывать данные из ПЛК
NumReg - число записываемых в абонент Z-регистров (1..123)
Cosv - признак косвеннной адресации буфера приема данных Должен быть 0 т.е допускается только прямая адресация
Buf - Массив - буфер приема данных из пассивного абонента.
Dgn - регистр диагностики линии связи, при запуске операции сюда записывается 0, по окончании операции сюда заносится ее результат (см. "Регистр диагностики сетевых функций ПЛК ReadNet, WriteNet")

TimeOut - Тайм-аут,мсек (50...65535)

5.3. Подпрограмма –
SnglWNet(Mesto:E_Word, Kanal:E_Word, Abonent:E_Word, BegRg:E_Word, Data:E_Word,

Dgn:V_Word, TimeOut:E_Word)

Запись через заданный канал связного модуля 52.05/52.07 одного регистра
данных в пассивный абонент.

Подпрограмма :
- запускает процесс записи (ModBus Func06)
- контролирует окончание записи или истечение тайм-аута на запись
- формирует в отведенном пользователем регистре диагностики линии связи результат операции

Параметры :
Mesto - номера места установки модуля в корзине ПЛК (0..15)
Kanal - номер канала в 52.05/52.07 (1 или 2)
Abonent - номер абонента в сети ModBus (1..31), подключенной к заданному каналу модуля 52.05/52.07
BegRg - номер Z-регистра в абоненте (0..65535), в который записать данные из ПЛК
Data - данные записывемые в пассивный абонент
Dgn - регистр диагностики линии связи, при запуске операции сюда записывается 0, по окончании операции сюда заносится ее результат (см. "Регистр диагностики сетевых функций ПЛК ReadNet, WriteNet")

TimeOut - Тайм-аут,мсек (50...65535)

5.4. Подпрограмма –
ActNet(Mesto:E_Word,Kanal:E_Word,Время:E_Word, Result:V_Word)

Проверяет активность канала связного модуля 52.05/52.07.

Функция сравнивает промежуток времени, прошедший с момента последнего
запуска операции в канале Канал связного модуля на месте Место, со временем Время.
При норме активности (промежуток менее заданной величины Время) возвращается Result = 1.
При ненорме активности (промежуток более заданной величины Время) возвращается Result = 0.

Цена младшего разряда параметра Время - 100ms, т.е. максимальное измеряемое время отсутствия активности 6553.500 сек = 109,2 мин

Если на указанном месте нет модуля 52.05/52.07,
то возвращается Result = 0.
Функция работает как для активных так и для пассивных каналов.

Для активных каналов время отмеряется от момента начала выполнения СФ ReadNet(WriteNet).
Для пассивных каналов время отмеряется от момента поступления запроса
от активной станции.

5.5. Подпрограмма –
DgnNet(Mesto:E_Word, Kanal:E_Word, Result:V_Word)

Подпрограмма DgnNet записывает в Result регистр сетевой диагностики для
канала Kanal связного модуля на месте Mesto главного каркаса.

Регистр сетевой диагностики содержит информацию о ненормах связи ПЛК (как пассивной станции так и активной станции), о ненормах обращения к сетевым функциям связи, и других ненормах, не вошедших в регистр диагностики связного модуля в функциях ReadNet, WriteNet.
После формирования слова диагностики сбрасывает внутренние данные регистра в соответствующем канале.

Структура регистра сетевой диагностики DgnNet одного канала связного модуля :

Бит	Описание
0	Резерв
1	"1" Тайм-аут при запуске ответа,- когда ПЛК-пассивная станция;
2	Резерв
3	"1" Отказ в запуске СпецФункции из-за некорректных параметров (в т.ч. на монтажном месте корзины, куда обращается СФ, нет 52.05/52.07)
4	"1" В активный канал поступили данные после отсчета тайм-аута
5	"1" Нарушение адресации сетей 52.05/52.07 (активная станция получает запросы или пассивная станция получает ответы)
6	"1" Подряд 2 блока информации по одному каналу в одном сеансе чтения из 52.05/52.07
7	"1" В активном канале ответ не соответствует запросу
8 ... 15	не используются

---

6. Сетевые процедуры ModBus TCP/IP

6.1. Подпрограмма - **DgnEthNet(Result:V_Word)**

Возвращает в Result регистр диагностики обмена ModBus TCP/IP/EtherNet,
после чего, если ненорма не относится к этапу инициализации Ethernet-контроллера RTL8100
(биты 0...3), сбрасывает этот регистр и бит DgnPLC._dpEthNet

Структура регистра диагностики обмена ModBus TCP/IP/EtherNet,
"1" в бите означает наличие ненормы(события) :

Бит 0 - в BIOS нет PCI функций
1 - Не найден Ethernet-контроллер Realtek 8100b
2 - Ненорма Irq Line отведенного BIOS\Plag&Play для Ethernet-контроллера
3 - За заданное время НЕ выполнился Reset 8100B
4 - резерв

5 - Отстыкован EtherNet-кабель или нет связи

6 - Переполнение очереди передатчика
7 - Резерв
8 - Переполнение Rx-буфера с наложением данных
9 - Переполнение списка сокет

10-15 - резерв

6.2. Подпрогрмм. - **ActEthNet(IPAdr:A2_Word, Время:E_Word, Result:V_Word)**

Проверяет активность заданного клиента для канала связи ModBus TCP/IP.

Функция сравнивает промежуток времени, прошедший с момента последнего
обращения заданного клиента к ПЛК(серверу), со временем *<Время>*.

Клиент задается своим IP-адресом, который располагается в массиве из 2x Z-слов - IPAdr.
Пример. IP-адрес 192.168.1.111 заносится в массив IPAdr следующим образом:

```
IPAdr[0]:=192+168*256
IPAdr[1]:=1+111*256
```

Цена младшего разряда параметра Время - 100ms, т.е. максимальное измеряемое время отсутствия активности 6553.500 сек = 109,2 мин

При норме активности (промежуток менее заданной величины Время) возвращается Result = 1.

При ненорме активности (клиент разорвал TCP соединения с ПЛК Более, чем “Время” назад или имеет TCP соединение, но промежуток более заданной величины “Время”) возвращается Result = 0.

Справка. "Клиент" - TCP аналог активной ModBus-станции
"Сервер" - TCP аналог пассивной ModBus-станции (т.е. ПЛК)

---

7. Процедуры над массивами

7.1. Подпрограмма - **CRC16Zone(Beg:V_any,LenW:E_Word,SCRC16:V_Word)**
Подсчитывает циклическую контрольную сумму CRC-16 стандарта ModBus (полиномиальное число A001h) заданной зоны памяти : начиная с адреса где расположен параметр Beg суммируются LenW слов.
Результат (слово) помещает в SCRC16.
Для сохранения надежностных характеристик CRC рекомендуется применять при объеме суммирования до 256 Z-слов.
LenW д.б. не более 32768 слов.

7.2. Подпрограмма - **CRC16ArrayW(Array:A_Word,SCRC16:V_Word)**
Подсчитывает циклическую контрольную сумму CRC-16 стандарта ModBus (полиномиальное число A001h) целого массива Array. Суммируются все элементы массива. Результат (слово) помещает в SCRC16.
Для сохранения надежностных характеристик CRC рекомендуется применять

при длине массива до 256 элементов.
Длина массива д.б. не более 32768 элементов.

7.3. Подпрограмма - **CRC16ArrayR(Array:A_Real,SCRC16:V_Word)**
Подсчитывает циклическую контрольную сумму CRC-16 стандарта ModBus (полиномиальное число A001h) вещественного массива Array.
Суммируются все элементы массива. Результат (слово) помещает в SCRC16.
Для сохранения надежностных характеристик CRC рекомендуется применять при длине массива до 128 элементов.
Длина массива д.б. не более 16384 элементов.

7.4. Подпрограмма - **CRC32Zone(Beg:V_any,LenW:E_Word,SCRC32:A2_Word)**
Подсчитывает циклическую контрольную сумму CRC-32
заданной зоны памяти : начиная с адреса где расположен параметр Beg суммируются LenW слов.
Результат (двойное слово) помещается в массив SCRC32 состоящий из двух элементов. Мадшие 16 бит CRC32 помещаются в нулевой элемент, старшие в первый.
Рекомендуется применять при объеме суммирования более 256 Z-слов.

7.5. Подпрограмма - **CRC32ArrayW(Array:A_Word,SCRC32:A2_Word)**
Подсчитывает циклическую контрольную сумму CRC-32
массива Array целых чисел. Суммируются все элементы массива. Результат (двойное слово) помещается в массив SCRC32 состоящий из двух элементов. Мадшие 16 бит CRC32 помещаются в нулевой элемент, старшие в первый.
Рекомендуется применять при объеме суммирования более 256 Z-слов.

7.6. Подпрограмма - **CRC32ArrayR(Array:A_Real,SCRC32:A2_Word)**
Подсчитывает циклическую контрольную сумму CRC-32
массива Array вещественных чисел. Суммируются все элементы массива. Результат (двойное слово) помещается в массив SCRC32 состоящий из двух элементов. Мадшие 16 бит CRC32 помещаются в нулевой элемент, старшие в первый.
Рекомендуется применять при объеме суммирования более 256 Z-слов.

7.7.**AvgMVxxx**

Подпрограмма подсчета скользящего среднего с автоматическим отведением буфера
хранения накопленных данных из отдельной зоны памяти вне ТД :
- AvgMV12 - Скользящее среднее 12b данных АЦП
- AvgMV16 - Скользящее среднее 16b данных

Подпрограмма подсчета скользящего среднего с "ручным" отведением буфера хранения
накопленных данных из зоны памяти ТД:
- AvgMV12t - Скользящее среднее 12b данных АЦП
- AvgMV16t - Скользящее среднее 16b данных

подпрограммы AvgMV12t, AvgMV16t предназначены для использования в подпрограммах

Общие сведения об усредняющих функциях:

подпрограмма AvgMVxxx реализует скользящее усреднение : удаляет самое "старое"
значение параметра, сохраняет новое значение параметра и вычисляет среднее.

Существуют 2 варианта подпрограмм скользящего усреднения AvgMV:
- AvgMV12x : для 12-битных данных АЦП с диагностичекими битами 15..12
и информационной частью в битах 11...0 (целое без знака),-
к вычисленному 12-битному среднему этих данных, после обработки, приформировываются биты 15..12 текущего значения
усредняемого параметра ;
- AvgMV16x : для 16-битных данных (целых без знака)

В процессе работы разрешается смена длины усредняемых данных.

Подпрограмма AvgMV поддерживает, в процессе работы, изменение базы усреднения
(числа усредняемых значений параметра) в пределах заданного максимального
числа точек, при этом, после каждой смены базы, соответствующим образом
учитываются ранее накопленные данные.

Максимальное значение усредняемых параметров фиксируется на этапе компиляции проекта (не может быть изменено в процессе работы функции).

Для хранения накопленных, при скользящем усреднении данных, выделяется
отдельная зона данных : S-зона данных усреднения объемом LenSZone слов.
Распределение S-зоны выполняет САПР,обрабатывая как AvgMV12 так и AvgMV16
Перераспределение S-зоны в процессе работы РП невозможно.
Непосредственное обращение к этим данным из РП невозможно.
Каждый отдельный вызов функции в САПР (даже с повторяющимися параметрами)
порождает отдельный канал усреднения - т.е. расход памяти S-зоны.

Длина S-зоны данных усреднения : LenSZone = 115232 слов.

Кроме того, пользователь может самостоятельно отводить зоны в ТД, при этом, пользователь должен :
- обеспечить перед началом усреднения обнуление первого слова в зоне усреднения
- не использовать зону усреднения для других целей пока эти данные используются для усреднения.

Один канал усреднения занимает (Nmax+8) слов, где Nmax - максимальное
количество усредняемых значений параметра.

При автоматическом распределении буфера хранения накопленных данных из отдельной зоны памяти вне ТД, каждый отдельный вызов функции в К748 (даже с повторяющимися параметрами) порождает отдельный канал усреднения.

При равномерном полном распределении памяти S-зоны между каналами усреднения максимально возможное количество точек усреднения в одном канале :

Nmax=[LenSZone/Nканалов]-8

Для 12*16 = 192 каналов усреднения Nmax = 592, т.е. время усреднения 592*0.05сек=29.6сек
Для (8*16-1)*12 каналов усреднения Nmax = 67 (67*0.05=3.35сек)

Формат подпрограмм при автоматическом распределении памяти S-зоны :
AvgMVxx(Prm:E_Word, N:E_Word, Nmax:C_Word, Result:V_Word)

Формат подпрограмм при явном (пользовательском) задании S-зоны :
AvgMVxxt(Prm:E_Word, N:E_Word, Nmax:C_Word,
Result:V_Word, Buffer:V_Word)

Буфер Buffer должен иметь размерность не менее (Nmax+8) слов.

Параметры подпрограмм :
Prm - текущее значение усредняемого параметра
N - текущее значение базы усреднения (числа усредняемых значений параметра)
Nmax - максимальная база усреднения
Result - рассчитанное среднее
Buffer – массив накопленных данных

Особые случаи:
а) N=0 : СФ выполняется при N=1
б) N>Nmax : СФ выполняется при N=Nmax
в) Последний адрес данных вне S-зоны(вне TD) :
Result=0 ; DgnRP2.SZonePrm=1
г) В начале буфера нет признака начала записи канала усреднения :
Result=0 ; DgnRP2.SZone=1
д) Изменился Nmax : Result=0 ; DgnRP2.SZonePrm=1

Описание слова диагностики РП DgnRP2 см.DgnPLCRg.

7.8. Подпрограмма - **MUXW(A:A_Word, Index:E_Word, WData:V_Word)**
Переносит элемент номер Index массива целых A в целую переменную WData.
Элементы нумеруются с 0.
Если заданный элемент вне массива, то возвращается последний элемент массива

7.9. Подпрограмма - **MUXR(A:A_Real,Index:E_Word, RData:V_Real)**
Переносит элемент номер Index массива вещественных A в вещественную переменную RData.
Элементы нумеруются с 0.
Если заданный элемент вне массива, то возвращается последний элемент

массива.

7.10. Подпрограмма - **ToArrayW(WData:E_Word, A:A_Word, Index:E_Word)**

Переносит значение целого выражения WData в элемент номер Index массива целых A.

Элементы нумеруются с 0.

Если заданный элемент вне массива, то ничего не выполняется

7.11. Подпрограмма - **ToArrayR(RData:E_Real, A:A_Real, Index:E_Word)**

Переносит значение вещественного выражения RData в элемент номер Index массива
вещественных A.

Элементы нумеруются с 0.

Если заданный элемент вне массива, то ничего не выполняется

7.12. Подпрограмма - **MaxW(A:A_Word, WData:V_Word)**

Ищет максимальный элемент WData массива целых без знака A

7.13. Подпрограмма - **MaxInt(A:A_Int, IData:V_Int)**

Ищет максимальный элемент IData массива целых со знаком A

7.14. Подпрограмма - **MaxR(A:A_Real, RData:V_Real)**

Ищет максимальный элемент RData массива вещественных A

7.15. Подпрограмма - **MinW(A:A_Word, WData:V_Word)**

Ищет минимальный элемент WData массива целых без знака A

7.16. Подпрограмма - **MinInt(A:A_Int, IData:V_Int)**

Ищет минимальный элемент IData массива целых со знаком A

7.17. Подпрограмма - **MinR(A:A_Real, RData:V_Real)**

Ищет минимальный элемент RData массива вещественных A

7.18. Подпрограмма - **FillArrayW(A:A_Word, Code:E_Word)**

Записывает во все элементы массива целых A одно и тоже
слово-заполнитель Code.

7.19. Подпрограмма - **FillArrayR(A:A_Real, Code:E_Real)**

Записывает во все элементы массива вещественных A одно и тоже
вещественное число-заполнитель Code.

7.20. Подпрограмма - **FillZoneW(Beg:V_Any, Len:E_Word, Code:E_Word)**

Записывает во все элементы зоны начинающейся с адреса идентификатора “Beg” одно и тоже слово-заполнитель Code.

Длина зоны Len - количество целых чисел в зоне

Настоящая функция инициализирует данные только согласно заданной длины зоны - при этом фактическая структура зоны ИГНОРИРУЕТСЯ - поэтому следует внимательно расчитывать Len, т.е. следить за тем какие переменные входят в зону

---

8. Процедуры работы с реальным (астрономическим) временем

8.1. **Общие сведения об обработке данных реального времени в ПЛК59.05pm**

8.1.1. Основные возможности реализуемые МО для часов реального времени
- два формата записей(наборов) данных :
  -- Запись DU : для измерения продолжительности промежутка времени (дискретность - 1 сек)
  -- Запись DT :
  --- для фиксации абсолютных моментов времени (дискретность - 0.001 сек)
  --- для сравнения двух абсолютных моментов времени (дискретность - 1 сек)
  --- для расчета промежутка времени между двумя абсолютными моментами времени в сек (дискретность - 0.001 сек)
- набор подпрограмм, позволяющий сразу целиком оперировать с записями времени без обработки их составляющих частей
- проверка работоспособности часов
- установка заданного времени дни и даты по команде извне ПЛК
- получение текущих времени дни и даты
- получение дополнительных данных : номера дня недели, признака "летнее время"
- автопереход на летнее время европейского стандарта и обратно, автоматическое определение периода врмени (стандартное(зимнее)/летнее)
- часы реального времени и тактовая сетка ПЛК не синхронизированы.

8.1.2. Формат записи "Абсолютные данные реального времени" - "DT" запись (Data&Time запись)

Этот формат хранит данные об абсолютном моменте времени.

В одной записи всегда хранятся как дата так и время суток, в Z-памяти они располагаются подряд, каждый параметр занимает одно слово (два b), что позволяет легко изменять те или иные поля записи

Одна DT-запись имеет длину 9 слов (18b).
В памяти следует резервировать место только под полную запись.
Ниже, этот формат записи обозначен как "DT".

Каждый параметр установлен в слове справа - младший бит параметра является битом 0 Z-слова.

Структура DT записи :

OfsZ	Параметр
0	Управление

Набор бит для определения состояния часов и управления ими.

Структура параметра "Управление" :

Бит	15	14	13	12..8	7..1	0
	S	WST				Set

Бит0 - Set : команда "Установить время или время и дату"
только запись "1"
Для запуска установки времени или времени и даты
следует занести соответствующие данные в поля записи,
установить этот бит в 1, и вызвать подпрограмму
SetTOD_T или SetDT_T.
После установки это бит автоматически сбрасывается в
"0"
и последующие вызовы подпрограммы SetTOD_T или SetDT_T
игнорируются

Биты 13...1 резерв

Бит14 - WST : признак текущего периода времени (Winter/Summer
Time)
только чтение
0 период стандартного времени (зимнего времени)
1 период летнего времени
Если отключен учет летнего времени, то этот признак
постоянно равен 0

Бит15 - S : текущее состояние часов (Status)
только чтение
0 часы функционируют или не включены в конфигурации
проекта
1 часы не функционируют (отсутствуют, неисправны,
разряжена батарея питания часов)
Этот бит обновляется каждую секунду абсолютного
времени
Этот бит дублируется в бит DgnPLC.RTC (см. СФ DgnPLC)

1	Мсек	миллисекунды 0..999
2	Сек	секунды,0..59
3	Мин	минуты,0..59
4	Час	биты 0..7 - часы,0..23 бит 8 "1" - время принадлежит повторному часу при переходе на стандартное время 0 - остальные варианты времени (т.е. этот бит взводится на 1час в году) биты 15..9 всегда равны 0
5	ДеньН	день недели 1..7 (1-понедельник,...,7-воскресенье)
6	ДеньМ	день месяца,1...31
7	Мес	месяц,1..12
8	Год	год от Рождества Христова (для 2004г здесь хранится число 2004)

Примечания.
- поле "МСек" предназначено только для :
-- различения сканов в пределах одной секунды реального времени
в системах документирования
-- расчета промежутка времени между 2мя абсолютными моментами
времени;

т.е. это поле игнорируется во всех СФ работающих с реальным временем,
кроме подпрограмм GetTOD, GetDT, SecDTR

- после запуска ПЛК, в течение времени не более 1 секунды, значение поля Мсек увеличивается от скана к скану,но не соответствуют реально прошедшему времени после начала первой астрономической секунды работы ПЛК; при первой же смене астрономических секунд, после запуска ПЛК, поле МСек уже нормально соответствует времени прошедшему после начала текущей секунды.
- в сутки перехода на летнее время после 1ч 59мин 59сек следуют 3ч 00мин 00сек
- в сутки перехода на стандартное (зимнее) время, после первых 2ч 59мин 59сек следуют 2ч 00мин 00сек т.е. в этих сутках интервал 2ч 00мин 00сек ... 2ч 59мин 59сек повторяется дважды, - бит Час.8 устанавливается в "1" только для второго такого интервала, весь остальной год он равен 0.
  МО ПЛК автоматически учитывает этот бит.
  Внешние устройства должны при расчетах промежутков времени учитывать возможность установки этого бита в "1", а при создании баз данных введение этого бита в индекс позволяет исключить ложное дублирование записей для "повторного часа"

8.1.3. Формат записи "Промежуток времени" - DU (time DUration) запись

Этот формат хранит данные о промежутке времени в диапазоне от 1сек до 60 лет.

Данные одной записи располагаются в Z-памяти подряд, каждый параметр занимает одно слово.

Одна DU-запись имеет длину 4 слова (8b).
В памяти следует резервировать место только под полную запись.

Ниже,этот формат записи обозначен как "DU".

Каждый параметр установлен в слове справа - младший бит параметра является битом 0 Z-слова.

OfsZ	Параметр	
0	Сек	секунды,0..65535
1	Мин	минуты, 0..65535
2	Час	часы, 0..65535
3	Дни	дни, 0..65535

Примечания

1. Поля записи допускают задание величин превышающих ограничения стандартной сетки времени :
   промежуток в 7 дней и 7 секунд м.б. задан как :
   (Дни=7,Час=0, Мин=0,Сек=7) или
   (Дни=0,Час=168,Мин=0,Сек=7) или
   (Дни=6,Час=24, Мин=0,Сек=7)
2. Общее заданное время д.б. не более 60 лет

8.1.4. Формат регистра "Результат сравнения DT/DU"

В этом формате выдаются результаты СФ сравнения записей абсолютных данных (DT) и записей промежутков времени(DU).
"A" - означает первый(левый) параметр в команде сравнения.
Данные этого формата занимают одно слово.

Бит	Имя		Назначение
0...7			резерв
8	NC	1	Данные несравнимы (ошибка в данных)
9	EQ	1	A=B
10	NE	1	A<>B
11	LT	1	A<B
12	LE	1	A<=B
13	GT	1	A>B
14	GE	1	A>=B
15	резерв		

Если бит равен "1", то соответствующее соотношение выполнено.
Одновременно устанавливаются биты ВСЕХ выполненных соотношений.

8.1.5. Переход на летнее время ("Day Light Saving")

ПЛК производит переход со стандартного времени на летнее и обратно согласно европейского стандарта.
Часовые пояса : Украина=+2, Белоруссия=+2, Россия=+3.

Ниже времена приведены для "+2"го часового пояса.
В последнее воскресенье марта после 1ч 59мин 59сек следуют 3ч 00мин 00сек
и наступает летнее время (сутки длятся 23 часа)

В последнее воскресенье октября после первых 2ч 59мин 59сек следуют 2ч 00мин 00сек и наступает стнадартное (зимнее) время, т.е. эти сутки длятся 25 часов и в них есть два промежутка 2ч 00м 00с .... 2ч 59мин 59сек (летний и зимний)

Поддерживает ли ПЛК переход на летнее время и обратно задается в конфигурации проекта.

Если время в ПЛК периодически устанавливается извне, и внешнее звено учитывает летнее время, то и в ПЛК так же следует включить автопереход
на летнее время - для правильного вычисления промежутков времени включающих точку смены времени при переходах летнее->зимнее время или обратно.

При расчете промежутков времени признак летнее/зимнее время учитывается
в ПЛК автоматически.

Для верного учета летнего времени при индексирования баз данных, куда
входит DT-запись "дата/время", следует включать в индекс бит Hour.8
который сделает уникальными записи повторяющегося промежутка 2..3 часа
при переходе на стандартное время.

Информация о текущем периоде (летний/стандартный) содержится в бите WST слова DT.Control, которое можно получить через СФ GetTOD/GetDT.

8.1.6. Примеры

Примечание : В примерах, приведенных ниже, обеспечение разового выполнения операции не отражено.

а) Каждый день в 11час 21мин 17сек выполнять действие A

```
Блок_i
 SetConstDT(65535,65535,65535,65535,11,21,17,RDT1);

Блок_j
 GetTOD(RDT2);
 CMPDT(RTD1,RDT2,Res);
 If Res.0 = True then   (*  Res.0 это RDT1 EQ RTD2 *)
    { Выполнить A; Res=0 };
 End_If;
```

б) Каждую пятницу 13го выполнять действие A

```
Блок_i
 SetConstDT(65535,65535,65535,5,65535,65535,65535,RDT1);

Блок_j
 GetTOD(RDT2);
 CMPDT(RTD1,RDT2,Res); (*; Res.0 это RDT1 EQ RTD2 *)
 If Res.0=1 then
   { Выполнить A; Res=0 };
 End_If;
```

Блок_j должен гарантированно вызываться хотя бы один раз в секунду - иначе необходимо применить соотношение GE

в) Каждые 1час 2мин 3сек выполнять действие A

```
 Блок_0 (инициализация)
 SetConstDU(0,1,2,3,ZDTRecC);
 GetDT(ZDTRecA);

 Блок_i (постоянно выполняемый)
```

```
CmpTempDU(ZDTRecA,ZDTRecC,Z600);
If Z600.0=1 then
   { GetDT(ZDTRecA);
     Выполнить действие A }
End_If;
```

8.2 Подпрограмма - **GetTOD(DT:A9_Word)**
Подпрограмма фиксирует в записи DT текущее время суток на момент начала скана, где ее вызывают. Все поля даты устанавливаются в 0. Вырабатывается бит Управление.S.

8.3 Подпрограмма - **GetDT(DT:A9_Word)**
Подпрограмма фиксирует в записи DT текущие дату и время суток на момент начала скана, где ее вызывают. Заносится бит Управление.S.

8.4 Подпрограмма - **SetTOD(DT:A9_Word)**
Подпрограмма устанавливает новое время суток из записи DT.
Поле МСек записи DT игнорируется.
Поля даты записи DT игнорируются.
Установка выполняется если бит Управление.Set равен "1".
Если бит Управление.Set равен "0", то функция ничего не выполняет.
В процессе установки нового времени бит Управление.Set автоматически сбрасывается в ноль и заносится бит Управление.S.
Установка времени производится непосредственно в процессе выполнения функции. Вызовы GetTOD или GetDT, произведенные после вызова SetTOD в том же скане, вернут новое время суток.
Если хотя бы одна из составляющих времени некорректна, то установка времени не выполняется и бит DgnRP2.RTCData устанавливается в "1".

8.5 Подпрограмма - **SetDate(DT:A9_Word)**
Подпрограмма устанавливает новую дату из записи DT.
Поля текущего времени записи RDT игнорируется.
Поле записи DT.ДеньН (день недели) игнорируется.
Установка выполняется если бит Управление.Set равен "1".
Если бит Управление.Set равен "0", то функция ничего не выполняет.
В процессе установки нового времени бит Управление.Set автоматически сбрасывается в ноль и заносится бит Управление.S.
Установка даты производится непосредственно в процессе выполнения функции. Вызовы GetTOD или GetDT, произведенные после вызова SetTOD в том же скане, вернут новую дату.
Если хотя бы одна из составляющих даты некорректна, то установка даты не выполняется и бит DgnRP2.RTCData устанавливается в "1".

8.6 Подпрограмма - **SetDT(DT:A9_Word)**
Подпрограмма устанавливает новые дату и время суток из записи DT.
Поле МСек записи RDT игнорируется.
Поле записи DT.ДеньН (день недели) игнорируется.
Установка выполняется если бит Управление.Set равен "1".
Если бит Управление.Set равен "0", то функция ничего не выполняет.
В процессе установки нового времени бит Управление.Set автоматически сбрасывается в ноль и заносится бит Управление.S.
Установка даты и времени производится непосредственно в процессе выполнения функции. Вызовы GetTOD или GetDT, произведенные после вызова SetTOD в том же скане, вернут новые дату и время суток.

Если хотя бы одна из составляющих даты или времени некорректна, то установка даты, времени не выполняется и бит DgnRP2.RTCData устанавливается в "1".

8.7 **SetConstDT(Год:E_Word,Месяц:E_Word,День:E_Word,**
ДеньНедели:E_Word,Часы:E_Word,
Минуты:E_Word,Секунды: :E_Word,
DT:A9_Word)

Заносит заданные по полям дату и время в запись DT, вычисляет при этом День Недели (RDT.ДеньН) и признак летнего времени (RDT.Управление.WST).
Поле RDT.MCек устанавливается в 0.

Поддерживает маскирование полей записи DT.
См. описание записи типа DT.

Допустимые значения :
Год = 1980,1981,...
Месяц = 1..12
День = 1..Число дней в месяце Месяц
ДеньНедели = 1..7
Часы = 0..23
Минуты = 0..59
Секунды =0..59

Допускается так же значение любого поля равное FFFFh(65535) - оно маскирует(исключает) это поле в сравнениях при выполнении спецфунцкии сравнения CMPDT.

Если в одном из полей задано недопустимое значение, то в DT заносится запись {1980.01.01,вторник, 0час 0мин 0сек} и устанавливается бит DgnRP2.RTCData.
Например : дата 2001.11.31 является ошибочной.

Если Год, Месяц, День не замаскированы, то в поле ДеньНедели записи RTD заносится не заданное, а рассчитанное функцией значение соответствующее заданной дате - что следует учитывать при задании масок.

В параметре RDT.Управление бит WST устанавливается согласно дате и часов времени (если они не замаскированы),остальные биты обнуляются.

Примечание. Если установлена поддержка летнего времени, то :
- если задано время из промежутка 2ч00мин00сек ... 2ч59мин59сек в последнее воскресенье марта (ошибка пользователя), то принимается одно и то же время 3ч00м00сек;
- для заданного промежутка 2ч00мин00сек ... 2ч59мин59сек в последнее воскресенье октября устанавливается признак летнего времени;

8.8 Подпрограмма - **CMPDT(DT_A:A9_Word;DT_B:A9_Word;Res:V_Word)**
Сравнивает две абсолютные точки (записи) даты и времени DT_A и DT_B. Если одно из составляющих полей любой из записей равно FFFFh(65535) (замаскировано), то операция сравнения для этого поля не выполняется.

Поля Control не сравнивается.
Поля МСек не сравнивается.
Возвращает результат в формате регистра "Результат сравнения DT/DU", который заносится в переменную Res.

8.9 Подпрограмма - **SecDTR(DT_1:A9_Word; DT_2:A9_Word; Secs:V_Word)**
Вычисляет промежуток времени в секундах между двумя абсолютными точками времени Secs=DT_2-DT_1.
Поля МСек используются в расчетах (точность измерения - 1мсек без учета использования данных времени на момент начала скана)
Результат - вещественное.
Если одно из полей в одной из записей недостоверно, то возвращается результат 0сек и устанавливается бит DgnRP2.RTCData.

8.10 Подпрограмма - **SetConstDU(Дни:E_Word,Часы:E_Word,**
Минуты:E_Word,Секунды:E_Word,
DU:A4_Word)
Заносит заданный промежуток времени в запись DU.
Если задан промежуток более 60лет,то в DU заносится значение 60*365 дней и устанавливается бит DgnRP2.RTCData.
См. описание записи типа DU.

8.11 Подпрограмма -
CMPDU(DT_1:A9_Word,DT_2:A9_Word,DU:A4_Word,Res:V_Word)
Сравнивает промежуток времени между двумя абсолютными точками времени (DT_2-DT_1) с промежутком времени RDU.
Поля МСек не сравниваются.
Возвращает результат в формате регистра "Результат сравнения DT/DU", который заносится в переменную Res.

8.12 Подпрограмма - **CMPTempDU(DT_1:A9_Word, DU:A4_Word, Res:V_Word)**
Сравнивает промежуток времени между абсолютной точкой времени DT_1 и текущей абсолютной точкой времени с промежутком времени DU.
Поля МСек НЕ сравниваются.
Возвращает результат в формате регистра "Результат сравнения DT/DU", который заносится в переменную Res.

---

9. Разные процедуры

9.1. Подпрограмма - **Delay(N:E_Word)**
Выполняет задержку на N мсек, т.е. время скана ПЛК, в котором, выполняется эта СФ будет, увеличено на N мсек.
Если N=0 то нет задержки. Точность отработки 0.5 мсек.

9.2. Подпрограмма - **ScanTime(Z:V_Word)**
Записывает длительность предыдущего скана ПЛК, в мсек, в переменную Z

9.3. Подпрограмма - **ReadRgMod(Mesto:E_Word,OfsUnit:E_Word,RdData:V_Word)**

Подпрограмма непосредственного чтения данных из одного или 2х восьми битных
регистров модуля главного каркаса, чтение выполняется немедленно - в точке вызова Подпрограммы.

Mesto - Место модуля в каркасе (0..15)
OfsUnit - Код читаемого набора регистров в модуле (0...7 - см.ниже)
RdData - результат чтения (слово)

Ofs Unit	Что читать	РезультатLo	РезультатHi
0	Rg0,Rg1	Rg0	Rg1
2	Rg2,Rg3	Rg2	Rg3
4	Rg4,Rg5	Rg4	Rg5
6	Rg6,Rg7	Rg6	Rg7
1	Rg1	Rg1	0
3	Rg3	Rg3	0
5	Rg5	Rg5	0
7	Rg7	Rg7	0

При ненорме чтения данных производится 3 повтора чтения - затем выставляется признак ОВВ для указанного места, при этом RdData не изменяется.

Если нет сильных побуждений, рекомендуется использовать для чтения модулей системные процедуры (т.е. просто указывать модули в конфигурации проекта)

9.4. Подпрограмма -
WriteRgMod(Mesto:E_Word,OfsUnit:E_Word,WrData:E_Word)

Подпрограмма непосредственной записи данных в один или два восьми битных регистра модуля главного каркаса ,запись выполняется немедленно - в точке вызова подпрограммы.

Mesto - Место модуля в каркасе (0..15)
OfsUnit - Код записываемого набора регистров в модуле (0...7 - см.ниже)
WrData - записываемые данные (слово или байт в Lo)

Ofs Unit	Куда писать	Данные Lo	Данные Hi
0	Rg0,Rg1	Rg0	Rg1
2	Rg2,Rg3	Rg2	Rg3
4	Rg4,Rg5	Rg4	Rg5
6	Rg6,Rg7	Rg6	Rg7
1	Rg1	Rg1	-
3	Rg3	Rg3	-
5	Rg5	Rg5	-
7	Rg7	Rg7	-

При ненорме записи данных производится 3 повтора записи - затем выставляется признак ОВВ для указанного места

Если модуль, в который записываются данные, указан в конфигурации проекта, то системные процедуры, еще раз запишут в него данные,

по окончании РП, но уже из системного буфера.

Если нет сильных побуждений, рекомендуется использовать для записи модулей системные процедуры (т.е. просто указывать модули в конфигурации проекта)

9.5. Подпрограмма - **FixMSec(BegTime:A2_Word)**

Фиксирует значение системного таймера ПЛК в массиве BegTime.
Время берется на момент начала скана ПЛК, в котором вызывается настоящая подпрограмма.
Таймер считает: 0,1, ... 655359,0,1,... .
Цена младшего разряда - 1мсек.
Размер массива BegTime - 2 слова.

9.6. **CmpTempMSec(BegTime:A2_Word, T:E_Real , RgCmp:V_Word)**

Сравнивает промежуток времени между ранее зафиксированной точкой времени BegTime и текущей точкой времени со значением T.
Результат заносится в переменную RgCmp.
Результат имеет формат регистра "Результат сравнения MSec" (см.ниже).
Время измеряется по системному таймеру ПЛК на момент начала скана.
Дискрета измерения времени - 1мсек.
Максимально измеряемый промежуток времени 655359 мсек (около 11 мин)
Точность измерения : -0.0016%.
T задается в мсек. Используется только целая часть T.
Размер массива BegTime - 2 слова.

Примечание Функции FixMSec, CmpTempMSec предназначены для измерения промежутков времени до 11 мин с дискретностью 1мсек (с учетом фиксации времени на момент начала скана).
Пример применения :
- разово фиксирутся момент, соответствующий какому-то событию с помощью FixMSec
- постоянно вызывается функция CmpTempMSec с заданным интервалом T пока RgCmp не покажет истечение интервала **T**

. Формат регистра "Результат сравнения MSec" :

Бит	Имя		Назначение
0...7			резерв
8	NC	1	Данные несравнимы (ошибка в данных)
9	EQ	1	Промежуток=T
10	NE	1	Промежуток<>T
11	LT	1	Промежуток<T
12	LE	1	Промежуток<=T
13	GT	1	Промежуток>T
14	GE	1	Промежуток>=T
15	резерв		

9.7. Подпрограмма - **OffADCSensor(Mesto:E_Word)**

Отключает, со следующего скана ПЛК, датчики ВСЕХ каналов модуля АЦП главного каркаса, находящегося на месте **Место**, от входных схем модуля АЦП.

Пояснения :

Модули АЦП, при измерении, пропускают через датчик импульсный ток. Поэтому, при одновременной работе нескольких модулей АЦП от одного датчика, модули АЦП могут выдавать искаженную информацию за счет наложения измерительных импульсов.

Функции OffADCSensor и OnADCSensor предназначены для организации дублированного использования нескольких модулей АЦП при работе от одного датчика - с их помощью пользователь может самостоятельно выбирать ведущий модуль АЦП.

При программировании следует учитывать :

- не все модули АЦП поддерживают описанную возможность - для точного определения этого следует обратиться к технической документации на модуль; применение функции OffADCSensor к модулю АЦП, не поддерживающему программное отключение измерительного датчика, приводит к искажению данных из модуля.
- информация из системы "два модуля АЦП работающие от одного датчика" достоверна только тогда, когда к датчику подключен один модуль
- при включении ПЛК автоматически подключает КАЖДЫЙ модуль АЦП к датчику (т.е. в системе "два модуля АЦП работающие от одного датчика" к датчику подключены ОБА модуля)
- появление достоверной информации на выходе модуля АЦП гарантируется только спустя 600 мсек от момента подключения датчика.
- программное отключение от датчика равносильно состоянию "Обрыв"

9.8. **OnADCSensor(Mesto:E_Word)**

Подключает, со следующего скана ПЛК, датчики ВСЕХ каналов модуля АЦП главного каркаса, находящегося на месте Место, ко входным схемам модуля АЦП.

Подробности - смотри описание подпрограммы OffADCSensor.

9.5. **PIDP1** - пропорциональный интегро-дифференциальный регулятор

9.5.1 Подпрограмма PIDP1 реализует пропорциональный интегро-дифференциальный регулятор (ПИД) со следующими возможностями :

- системным рассогласованием, сводимым к нулю, является разность между уставкой Sp и контролируемым параметром Pt
- СФ, по текущему расогласованию и истории процесса, вычисляет выходной

параметра регулятора как сумму пропорциональной, интегральной и дифференциальной составляющих
- выбор для дифференциальной составляющей в качестве входного параметра системного рассогласования или текущего изменения входного параметра
- каждая из трех составляющих м.б. отдельно отключена (установкой Kp=0, Ki=0, Kd=0)
- режим "перезарядка" для интегральной составляющей
- ограничение раскачивающего действия интегральной компоненты
- ограничение выходного сигнала на задаваемых уровнях и фиксация этого события (выхода регулятора на "упоры") в специальных выходных битах
- сброс пользователем истории регулятора в процессе работы
- три режима работы :
  -- автоматический (основной)
  -- ручной
  -- останов
- плавный переход от ручного режима к автоматическому
- диагностика ошибок входных параметров и неизменение выходного параметра при их обнаружении
- отдельная память данных для контуров регулирования - вне ТД, максимальное число контуров регулирования - 500
- формирование копии текущих данных регулятора для его системной настройки
- для расчета временных промежутков используется время на момент начала скана ПЛК, дискретность измерения времени - 1мсек

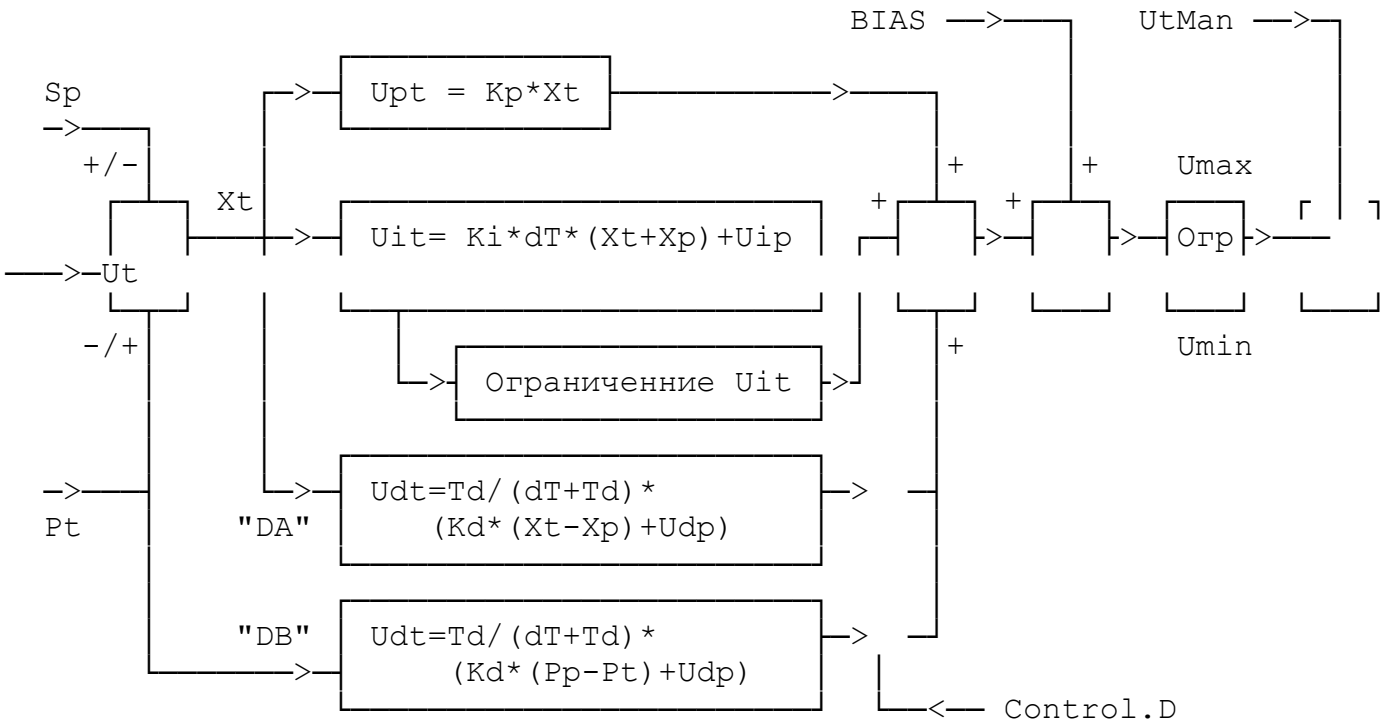
9.5.2 Вызов подпрограммы PIDP1

```
PIDP1 (
       Control : V_Word ,  (* регистр управления *)
       Контур  : E_Word ,  (* номер контура (0...499) *)
       Pt      : E_Real ,  (* текущее значение контролируемого параметра *)
       Revers  : E_Word ,  (* реверс - признак определяющий знак отклонения
                           текущего значения параметра Pt от уставки Sp :
                           Если Revers=0, то Xt = Sp-Pt
                           Если Revers=1, то Xt = Pt-Sp *)
        Kp      : E_Real , (* коэффициент пропорциональности (Kp>=0) *)
        Ki      : E_Real , (* интегральный коэффициент       (Ki>=0) *)
        Kd      : E_Real , (* дифференциальный коэффициент   (Kd>=0) *)
        Td      : E_Real , (* время запаздывания дифференциальной компоненты,
                              1ед=1сек *)
        Umin    : E_Real , (* минимальное значение выходного параметра *)
        Umax    : E_Real , (* максимальное значение выходного параметра *)
        BIAS    : E_Real , (* вход смещения *)
        Sp      : E_Real , (* уставка контролируемого параметра *)
        Utman   : E_Real , (* заданное значение выходного параметра Ut
                              для режима Manual *)
```

```
        Ut      : V_Real, (* текущее значение выходного параметра;
                             выход всегда является не особым
вещественным числом *)
        Data    : A37_Word,(* массив копии данных регулятора
                           (для системной настройки регулятора)
                            Длина массива 37 слов

                      Порядок отведения массива :
                      а) массив отводится в фиксированных адресах
                      начиная с 1го адреса распределяются не менее
                             37 слов согласно структуре массива :
                                  Ofs
                                   0     aControl       (целое)
                                   1     aКонтур
                                  ...
                                  35     aUwind_up_max  (вещественное)

                                Данные, в конце буфера, к которым доступ
                                из Динамики переменных  не нужен,
                                можно определить как некий массив с длиной
                                дополняющей до 37 слов.
                           б) в вызове функции для массива копии данных
                                регулятора указывается переменная Control*)

        )
```

Выходные параметры : Ut, данные в буфер копии внутренних данных
остальные - входные параметры

9.5.3 Упрощенная блок-схема ПИД регулятора



Обозначения:
- Xt - отклонение от уставки
- Upt - пропорциональная компонента
- Uit - интегральная компонента
- Udt - дифференциальная компонента

Uip,Udp,Xp,Pp - предыдущие значения соответствующих параметров

9.5.4 Структура входно-выходного параметра "Регистр управления Control"

```
15 14   13    12   11 10 9 8   7 6 5 4   3 21 0
 Z  E GtMax LtMin   -  - - -   - - D I   B MH R
```

Биты 5..0 являются входными битами
Биты 15..12 являются выходными битами

Бит 0 - R(Reset)
Если R="0", то нормальное функционирование : в текущем вызове
для расчетов берутся накопленные данные предыдущего вызова.
Если R="1", то выполняется сброс накопленных данных :
в текущем вызове накопленные данные контура предварительно
сбрасываются и эти сброшенные значения используются как
предыдущие при расчетах Ut.

Биты 2,1 - M,H (Manual, Halt)
Эти биты определяют режим работы регулятора согласно

следующей таблицы:

Бит2 MANUAL	Бит1 HALT	Режим регулятора
0	0	автоматический
0	1	стоп(halt)
1	0	ручной(manual)
1	1	ручной(manual)

Особенности режимов

а) Автоматический режим
Выход Ut определяется основным алгоритмом регулятора.
Выход ограничивается в дапазоне Umin ... Umax.
Производится ограничение раскачивающего действия интегральной компоненты.
В общем переход из автоматического режима в ручной происходит неплавно, т.к. возможно рассогласование между текущим Ut и Utman.

б) Ручной режим
Входной параметр Utman переносится в выходной параметр Ut,
при этом Ut ограничивается в диапазоне Umin ... Umax.
Внутренние переменные (Up,Ui) отслеживаются таким образом,
что контроллер м.б. плавно переключен из "ручного" режима
в автоматический; также выполняется ограничение раскачивающего действия интегральной компоненты.
В "ручном" режиме Ud автоматически устанавливается в 0

в) режим Стоп(Halt)
В режиме Halt параметр Ut сохраняет значение,которое имел
на момент перевода в этот режим.
Внутренние переменные отслеживаются таким образом, что что контроллер м.б. плавно переключен из режима Halt в автоматический; также выполняется ограничение раскачивающего действия интегральной компоненты.

Бит 3 - B(deBug)
Это бит включает копирование внутренних данных контура регулятора в заданный пользователем буфер, что м.б. использовано при исследовании поведения объекта регулирования.
Если бит D="0", то копирование не производится и в переменной
ZAdrBuf м.б. любое значение.
Если бит D="1", то после получения всех составляющих Ut, в буфер длиной 37 Z-слов, адрес которго задан в параметре ZAdrBuf (косвенная адресация), заносятся все входные параметры
контура, время между двумя вызовами СФ, все компоненты Ut и сам

параметр Ut.
Структура буфера копии данных регулятора копии приведена ниже.

Бит 4 - I(I-Portion)
Этот бит изменяет алгоритм вычисления интегральной составляющей выходного параметра в автоматическом режиме
Если бит I=0, то интегральная составляющая вычисляется согласно штатному алгоритму PIDP1.
Если бит I=1, то интегральная составляющая вычисляется в режиме "перезарядка" - без использования предыдущего значения интегральной составляющей (Uip=0) и предыдущего значения отклонения входного параметра от уставки (Xp=0), в остальном интегральная составляющая вычисляется как обычно (в т.ч. проводится ограничение ее раскачивающего действия)

Бит 5 - D(D-Portion)
Этот бит изменяет алгоритм вычисления дифференциальной составляющей выходного параметра в автоматическом режиме
Если бит D=0, то дифференциальная составляющая вычисляется согласно алгоритму DA (см. ниже).
Если бит D=1, то дифференциальная составляющая вычисляется согласно алгоритму DB (см. ниже).

Биты 5..11 резерв

Выходные биты СФ

Бит 12 - LtMin ("Значение Ut ограничено снизу")
0 - вычисленное значение Ut более Umin
1 - вычисленное значение Ut менее или равно Umin и было ограничено снизу : Ut=Umin

Бит 13 - GtMax ("Значение Ut ограничено сверху")
0 - вычисленное значение Ut менее Umax
1 - вычисленное значение Ut более или равно Umax и было ограничено сверху : Ut=Umax

Бит 14 - E(Error) Ошибка во входном параметре
0 - норма
1 - имеется ошибка во входном параметре
(см. ниже раздел "Особые случаи")

Бит 15 - Z(Zone) Разрушены данные контура регулятора
0 - норма
1 - разрушены данные контура регулятора

9.5.5 Алгоритм подпрограммы PIDP1

Контур регулирования состоит из трех компонент: пропорциональной, интегральной и дифференциальной. Все компоненты внутри функции вычисляются
в 32b виде в доп.коде.

Включение компонент в контур, определяется значением коэффициентов. Нулевое значение коэффициента Kp,Ki,Kd исключает соответствующую компоненту.

Отклонение X контролируемого параметра Pt от уставки Sp вызывает выработку ПИД-регулятором управляющего воздействия Ut:

Ut = PIDP1(Pt)

Алгоритм расчета Ut :

- Вычислить Xt : текущее отклонение входного Pt от уставки Sp
  Если Revers=0, то Xt = Sp-Pt
  Если Revers=1, то Xt = Pt-Sp

- Пропорциональная компонента : Upt = Kp*Xt

- Интегральная компонента Uit

  Если режим Halt, то :
  -- Uit= Utprev-Upt-BIAS
  -- перейти к ограничению раскачивающего действия интегральной компоненты

  Если режим Manual, то :
  -- Uit= Utman-Upt-BIAS
  -- перейти к ограничению раскачивающего действия интегральной компоненты

  Если Ki=0, то Uit=0 и окончить вычисление Uit

  Если режим вычисления Uit "перезарядка", то установить: Uip=0, Xp=0

  Вычислить i-компоненту : Uit= Ki*dT*(Xt+Xp)+Uip
  Примечание. В формуле для Uit отсутствует коэффициент "/2" - обеспечивающий "формулу трапеции" при интегрировании :
  в традициях Constar считать, что он входит в Ki.

  Ограничение раскачивающего действия интегральной компоненты:
  Если Uit>Umax-(Upt+BIAS), то Uit=Umax-(Upt+BIAS)
  Если Uit<Umin-(Upt+BIAS), то Uit=Umin-(Upt+BIAS)

- Дифференциальная компонента Udt

  Если режим Halt или Manual, то Udt=0 и окончить вычисление Udt.

  Если Kd=0, то Udt=0 и окончить вычисление Udt

  Вычислить K :
  Если dT=0, то { K=1.0 } иначе
  Если Td=0, то { Udt=0 и окончить вычисление Udt (K=0.0) } иначе
  K=1/(dT/Td+1)

Если Control.D=0 то AUdt=(Kd*(Xt-Xp)+Udp) (алгоритм DA)
Если Control.D=1 то AUdt=(Kd*(Pp-Pt)+Udp) (алгоритм DB)

Udt=AUdt*K

- Текущее значение выходного параметра
  Если режим Auto, то : Ut = Upt + Udt + Uit + BIAS
  Если режим Halt, то : Ut = Utprev
  Если режим Manual, то : Ut = Utman

- Ограничение значения выходного параметра
  Если Ut>Umax, то Ut=Umax
  Если Ut<Umin, то Ut=Umin

9.5.6 Структура буфера копии данных регулятора

Z-Смещение от (Data)	Тип данных	Данные
0	W	Control
1	W	Контур
2	R	Pt
4	W	Реверс
5	R	Kp
7	R	Ki
9	R	Kd
11	R	Td
13	R	Umin
15	R	Umax
17	R	BIAS
19	R	Sp
21	R	Utman
23	R	dT
25	R	Upt
27	R	Uit
29	R	Udt
31	R	Ut
33	R	Uwind_up_min : min-граница ограничения раскачивающего действия интегральной компоненты: Uwind_up_min=Umin-Upt-BIAS
35	R	Uwind_up_max : max-граница ограничения раскачивающего действия интегральной компоненты: Uwind_up_max=Umax-Upt-BIAS

Всего : 37 Z-слов

9.5.7 Особые случаи

- если адрес размещения выходного параметра Ut вне ТД,то :
  -- устанавливается бит Control.E

-- СФ ничего не выполняет
- Если задан несуществующий контур, то :
-- устанавливается бит Control.E
-- СФ ничего не выполняет
- Если Umin>Umax, то :
-- устанавливается бит Control.E
-- СФ ничего не выполняет
- Если Kp<0 или Ki<0 или Kd<0 то :
-- устанавливается бит Control.E
-- СФ ничего не выполняет
- Если разрушена запись контура, то :
-- устанавливается бит Control.Z
-- СФ ничего не выполняет
- если задан вывод в буфер системной отладки и последний адрес этого буфера вне ТД, то :
-- устанавливается бит Control.E
-- запись в буфер не производится

9.6. Подпрограмма **FFST(A:V_Word)**

(FlyFixSysTimer)

Чтение "на лету" сетчика системного таймера CPU (канала 0 системного таймера IBM) в целую без знака переменную A.

Счк таймера считает так : 11932 ...0, 11932... .
В переменную A заносится (11932-Счк),т.е A считает так : 0...11932 ,0 ...

Период 11932...0 соответствует 10ms.
Т.е. цмр A = 0.8380824673 мкс.

9.7. Поддержка протокола работы ПЛК в заданном пользователе массиве

Общая схема ведения протокола работы ПЛК :

- Протоколизация включается/отключается специальным флагом в паспорте проекта

- фиксируются следующие события из жизни ПЛК :
-- запуск ядра ПЛК
-- запуск РП ПЛК
-- останов РП ПЛК

- каждое событие пуска описывается записью следующей структуры из 7 слов:

Смещение в Z словах	Имя	Описание
0	Код События	01 Пуск ядра ПЛК 02 Пуск РП от тумблера 03 Пуск РП из К748 по команде Пуск

		04 Пуск РП из К748 по команде Скан
		10 Стоп ПЛК по обобщенному биту отказа
		20 Стоп ПЛК от команды СТП
		21 Стоп ПЛК из К748
1	Номер пуска ядра ПЛК	Текущий номер запуска ядра ПЛК 59.05 (то же, что возвращает и СФ PuskPLC) В младшем байте - Номер запуска (0,1..255,0...) Старший байт - резерв
2	Резерв	
3	МинСек	В младшем байте - Секунды наступления события (0..59) В старшем байте - Минуты наступления события (0..59)
4	ДеньЧас	В младшем байте - Час наступления события (0..23) В старшем байте - День наступления события (1..31)
5	Месяц	В младшем байте - Месяц наступления события (1..12) Старший байт - резерв
6	Год	Год наступления события

Примечание Если RTC отключены, то поля времени нулевые

- каждое событие останова описывается записью следующей структуры из 14 слов :

Смещение в Z словах	Имя	Описание
0 .. 6	данные как в записи для пуска	
7	DgnPLC	Общие признаки отказов ПЛК
8	DgnRP1	Отказы РП слово 1
9	DgnRP2	Отказы РП слово 2
10	DgnAdrRP	Адрес отказа РП (Low слово)
11	DgnAdrRP	Адрес отказа РП (High слово)
12	DgnPVC	Время скана при ПВЦ
13	DgnCMOS05	диагностика ненорм CMOS05

Примечание
Описание DgnPLC, ... DgnCMOS05 - см. файл Specfunc.doc, спец. функцию DgnPLCRg

- МО ПЛК, при наступлении события, автоматически помещает запись об этом событии в массив фиксированной длины 10*14 = 140 слов.

В массиве хранятся данные о последних 10..20 событиях в форме стека FIFO:

запись, расположенная в начале массиве, соответствует самому последнему событию, а запись, расположенная в конце массива, соответствует

самому раннему событию.

- массив располагается в ТД, его начальный адрес определяет пользователь в паспорте проекта - в двойном слове BegAdrProtokol Рекомендуется размещать этот массив в сохраняемой зоне ТД. Для получения данных из этого массива следует определить с этого адреса массив длиной 140 слова и применить "Динамику переменных".

  Т.о. пользователь должен только отвести место для массива протокола включить составление проткола и более ничего - об остальном позаботится система.

10. Процедуры для работы с адресами памяти

10.1 Функция - **ADDRESS(Переменная:V_ANY):Dword**

Возвращает адрес в байтах переменной указанной в качестве параметра.

В качестве параметра задается имя любой переменной, если это массив то необходимо указывать и индекс, но каково бы ни было значение индекса функция всегда вернет адрес нулевого элемента. Однако если в качестве переменной указать выход счетчика или таймера то она вернет адрес байта где хранятся выходы счетчиков или таймеров. Тип выходного значения **Dword, -** равное значению адреса указанной переменной в байтах.

10.2 Группа функций чтения информации по заданному адресу

1. **Byte_From_Addr(Адрес:E_Dword):Byte**
2. **Word_From_Addr(Адрес:E_Dword):Word**
3. **Dword_From_Addr(Адрес:E_Dword):Dword**
4. **Sint_From_Addr(Адрес:E_Dword):Sint**
5. **Int_From_Addr(Адрес:E_Dword):Int**
6. **Dint_From_Addr(Адрес:E_Dword):Dint**
7. **Real_From_Addr(Адрес:E_Dword):Real**

; где <Адрес> - выражение задающее адрес ячейки памяти в байтах. Функции возвращают содержимое соответствующих ячеек памяти таблицы данных. Тип возвращаемых значений соответствует названиям функций.

10.3 Группа подпрограмм записи информации по заданному адресу

1. **Byte_TO_Addr(Адрес:E_Dword, Val:E_Byte) :Byte**
2. **Word_TO_Addr(Адрес:E_Dword, Val:E_Word) :Word**
3. **Dword_TO_Addr(Адрес:E_Dword, Val:E_Dword):Dword**
4. **Sint_TO_Addr(Адрес:E_Dword, Val:E_Sint) :Sint**
5. **Int_TO_Addr(Адрес:E_Dword, Val:E_Int) :Int**
6. **Dint_TO_Addr(Адрес:E_Dword, Val:E_Dint) :Dint**
7. **Real_TO_Addr(Адрес:E_Dword, Val:E_Real) :Real**

; где <Адрес> - выражение задающее адрес ячейки памяти в байтах, <Val> - выражение определяющее значение которое будет записано в память с начальным адресом <Адрес> .Количество записываемых байт зависит от типа выражения <Val>.

10.4 Функция – **Dim(Переменная:V_ANY):Word**

Возвращает размерность (Dimension) переменной указанной в качестве параметра. В случае массивов возвращается число элементов в массиве, в случае скаляров возвращается единица. Имя массива должно записываться без указания индекса.

10.4 Функция – **Size(Переменная:V_ANY):Byte**

Возвращает длину переменной в байтах указанной в качестве параметра. Она однозначно определяется типом (разрядностью) переменной. Например для переменной типа Byte функция возвращает 1(один), Int – два, Real – четыре и т.п. Для переменной типа Bool (Дискретная) функция возвращает ноль. Если в качестве параметра задается массив то функция возвращает длину одиночного элемента, имя массива должно записываться без указания индекса.

11. Процедуры для работы с сегментами и блоками

11.1 Подпрограмма - **OPEN_SEGMENT(Сегмент:C_Byte)**
Открывает сегмент, указанный в качестве параметра. Оставляя в нем неизменным номер активного блока.

11.2 Подпрограмма - **CLOSE_SEGMENT(Сегмент:C_Byte)**
Закрывает сегмент, указанный в качестве параметра. Оставляя в нем неизменным номер активного блока.

11.3 Подпрограмма - **RESET_SEGMENT(Сегмент:C_Byte)**
Сбрасывает сегмент указанный в качестве параметра, т.е. устанавливает в сегменте активным нулевой блок, не изменяя состояние (открыт/закрыт) самого сегмента.

11.4 Подпрограмма - **GOTO_BLOCK(Блок:C_Byte)**
Устанавливает активным блок указанный в качестве параметра, не изменяя состояние открытости сегмента. Настоящая подпрограмма действует только на тот сегмент, в котором она вызывается.

11.5 Функция - **STATUS_SEGMENT(Сегмет: E_ Byte):Byte**
Возвращает байт состояния сегмента номер которого указывается в качестве параметра (может быть представлен произвольным выражением с типом "Byte").Внимание! Если значение выражения "Сегмент" превышает максимальный номер сегмента ошибка сформирована не будет.

Формат байта состояния сегмента:
Биты 0…6 – номер активного блока в рассматриваемом сегменте (0-127)
Бит 7 - Статус сегмента (0 – закрыт, 1 - Открыт)

12. Процедуры системы «горячего» резервирования (HSB)

12.1. Подпрограмма - **HSBNumPLC(AB: V_Word)** - монтажный номер ПЛК (A/B)
Возвращает в слово 'AB' тип перемычки на процессорном модуле резервированного ПЛК (A или B).
Значение 1 соответствует "A", 0 – "B"

12.2. Подпрограмма - **HSBStatus(Status: V_Word)** - Текущее состояние данного ПЛК(Ведущий/ Резерв/ OffLine)
Возвращает слово текущего состояния процессора в HSB системе.

Расшифровка слова Status:
0 – Состояние после подачи питания;
1 – Ожидание переключения HSB-тумблера в состояние "ПОДКЛ"
2 – Ожидание связи от мастера
3 – Ожидание ответа от подчиненного
4 – Состояние "Резерв"
5 – Состояние "Ведущий"
6 – Состояние "Дефект"

7 – Ожидание связи после исчезновения дефекта

12.3. Подпрограмма - **IntoStandBy()** - Уйти в "резерв"

При исполнении этой функции ведущий ПЛК передает управление ведомому а сам при этом уходит в "резерв".
Функция выполняется только при отсутствии противопоказаний к переходу, в противном случае функция игнорируется.

12.4 Подпрограмма - **HSBDgnPLC(A:V_Word)**

Получение обших данных о состоянии ПЛК партнера:
пересылает в переменную A основной регистр (слово) диагностики ПЛК парнера после чего сбрасывает бит Net в этом регистре.

Структура основного слова диагностики ПЛК партнера идентична слову DgnPLC (см. функцию **DgnPLC**)

12.5. Подпрограмма - **HSBDgnPLCRg(A:A10_Word)**

Записывает в массив A все регистры(слова) диагностики ПЛК партнера
Массив занимает 10 слов

Структура массива идентична массиву, используемому в функции **DgnPLCRg.**

12.6. Подпрограмма - **HSBUnits(Karkas:E_Word, Units:V_Word)**

Получение данных о расположении отказавшего модуля в ПЛК партнере:
пересылает в Units слово отказов модулей в главном каркасе или расширителе.

Karkas = 0 для главного каркаса
Karkas = 1..7 для каркасов расширителей

"1" в бите N слова Units означает ошибку ввода вывода (ОВВ) в модуле
на месте N (0...15)

12.7. Подпрограмма - **HSBDgnExp(A: V_Word)**

Пересылает в переменную A регистр наличия ошибок связи по интерфейсу RS485 всех расширителей связанных с ПЛК партнером.

Структура регистра :

Бит 0		Резерв
1	"1"	Имеются ошибки связи с расширителем 1
2	"1"	Имеются ошибки связи с расширителем 2
...		
7	"1"	Имеются ошибки связи с расширителем 7
8		Резерв
...		
15		Резерв

Если бит взведен, то произошла одна из ненорм :
- За контрольное время не начал поступать ответ из РАСШ при чтении

из него фрейма-данных
- Не совпала CRC при приеме фрейма (после 3х повторов)
- Данные "Из РАСШ" не соответствуют данным "В РАСШ" (после 3х повторов)
- Не совпала CRC при приеме фрейма в РАСШ (после 3х повторов)

Для уточнения причины установки бита - см. HSBDgnExpI.

12.8. Подпрограмма - **HSBDgnExpI(Karkas:E_Word, Units:V_Word)**

Пересылает в переменную DgnExpI регистр причины ошибки связи по интерфейсу RS485 ПЛК партнера с заданным расширителем Karkas (1..7)
Если задан номер каркаса 0 или более 7, то возвращается 0.

Структура регистра причины ошибки связи :

Бит 0 "1" После ответа из 2b не принят 3й BI -байт
1 "1" За контрольное время не начал поступать ответ из РАСШ при чтении из него фрейма-данных
2 "1" Ошибка данных в принятом из RXD байте (BI,FE,PE,OE)
3 "1" Не совпала CRC при приеме фрейма (после 3х повторов)
4 "1" Перерыв в поступлении байт фрейма более TSilence
5 "1" Поступил "чужой" адрес
6 "1" Поступил неверный код функции
7 Резерв
8 "1" Данные "Из РАСШ" не соответствуют данным "В РАСШ" (после 3х повторов)
9 "1" Не совпала CRC при приеме фрейма в РАСШ (после 3х повторов)

12.9. Подпрограмма - **HSBPuskPLC(NumberPusk:V_Word)**

Возвращает текущий номер запуска ядра ПЛК партнера в переменной NumberPusk
Номер хранится в энергонезависимой памяти.
Внешний наблюдатель, контролируя этот номер может фиксировать перезапуски ПЛК (ручные, от Watch Dog Timer'a);

Структура NumberPusk :
Биты 0..7 Номер запуска
Биты 8..15 Резерв