УТВЕРЖДЕН
ГУКН.421457.001 03 34 6206-1-ЛУ

КОМПЛЕКС ПРОГРАММНЫХ СРЕДСТВ
«КАСКАД-САУ»

Редактор алгоритмов IEC 1131-3

Руководство оператора

Приложение 3
Примеры программирования на языке FBD

ГУКН.421457.001 03 34 6206-4
Листов 17

2011

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

СОДЕРЖАНИЕ

1 ПРИМЕР 1: ПОЛУЧИТЬ СРЕДНЕЕ ИЗ ТРЕХ

Программа на языке FBD представлена на рисунке 3.1. Номера над блоками показывают, в каком порядке производится вычисление.

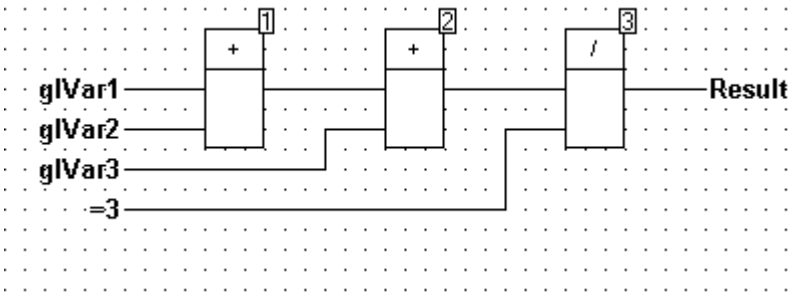


Рисунок 3.1

В данном примере не используются блоки и схемы, вычисления производятся непосредственно в теле проекта.

Описание программы представлено в таблице 3.1

Т а б л и ц а 3.1

Переменные	glVar1 : FLOAT, глобальная, индекс 0 glVar2 : FLOAT, глобальная, индекс 1 glVar3 : FLOAT, глобальная, индекс 2 Result : FLOAT, глобальная, индекс 3
Алгоритм	сумма переменных glVar1 … glVar3 делится на 3, результат помещается в переменную Result.
Код: точка входа на метке L2	01: (*#################### Begin of << Middle_3 >> *) 02: (*.............................. "+" *) 03: L2: FLoad regA, Vars[0] 04: FLoad regB, Vars[1] 05: FAdd regA, regB 06: FStore Vars[4], regA 07: (*.............................. "+" *) 08: L7: FLoad regA, Vars[4] 09: FLoad regB, Vars[2] 10: FAdd regA, regB 11: FStore Vars[5], regA 12: (*.............................. "/" *) 13: L12: FLoad regA, Vars[5] 14: FMov regB, 3 15: FDiv regA, regB 16: FStore Vars[6], regA 17: FStore Vars[3], regA 18: L17: Ret 19: (********************** End of << Middle_3 >> *)
Строка 1	Неисполняемая строка:

	комментарий, объявляющий начало кода проекта.
Строка 2	Неисполняемая строка: комментарий, объявляющий код для первого блока «+».
Строка 3	FLoad regA, Vars[0] Загрузить в регистр A FLOAT значение переменной glVar1 – первое слагаемое.
Строка 4	FLoad regB, Vars[1] Загрузить в регистр B FLOAT значение переменной glVar2 – второе слагаемое.
Строка 5	FAdd regA, regB Сложить FLOAT значения регистра A и регистра B, поместить результат в регистр A.
Строка 6	FStore Vars[4], regA Поместить результат сложения из регистра A в автоматическую переменную с индексом 4
Строка 7	Неисполняемая строка: комментарий, объявляющий код для второго блока «+»
Строка 8	FLoad regA, Vars[4] загрузить в регистр A FLOAT значение из автоматической переменной с индексом 4 – первое слагаемое.
Строка 9	FLoad regB, Vars[2] загрузить в регистр B FLOAT значение из переменной glVar3 – второе слагаемое.
Строка 10	FAdd regA, regB сложить FLOAT значения регистров A и B, результат поместить в регистр A.
Строка 11	FStore Vars[5], regA поместить результат сложения из регистра A в автоматическую переменную с индексом 5.
Строка 12	Неисполняемая строка: комментарий, объявляющий код для блока «/»
Строка 13	FLoad regA, Vars[5] загрузить в регистр A FLOAT значение автоматической переменной с индексом 5 – делимое.
Строка 14	FMov regB, 3 занести в регистр B FLOAT константу 3 – делитель.
Строка 15	FDiv regA, regB разделить FLOAT значение регистра A на FLOAT значение

	регистра B, результат поместить в регистр A.
Строка 16	FStore Vars[6], regA поместить результат деления из регистра A в автоматическую переменную с индексом 6.
Строка 17	FStore Vars[3], regA поместить результат деления в переменную Result.
Строка 18	Ret завершить исполнение проекта.
Строка 19	Неисполняемая строка: комментарий, объявляющий конец кода проекта

2 ПРИМЕР 2: ОРГАНИЗАЦИЯ ЦИКЛА С ПОСТ-ПРОВЕРКОЙ ЧИСЛА ИТЕРАЦИЙ

Программа на языке FBD представлена на рисунке 3.2. Номера над блоками означают порядок исполнения.

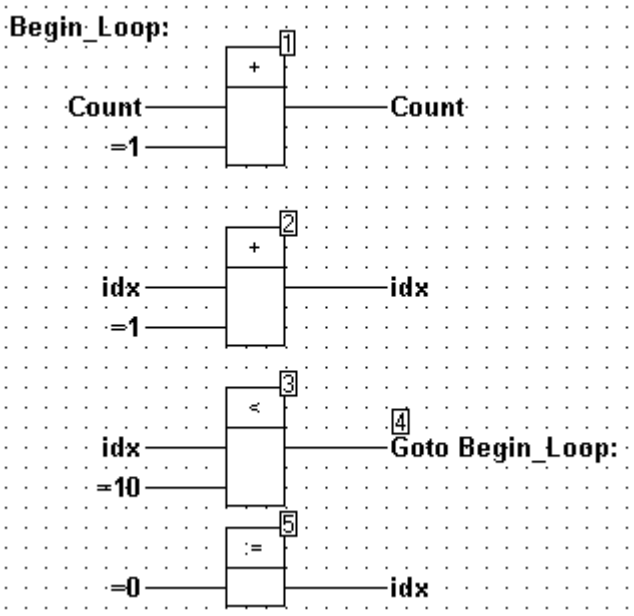


Рисунок 3.2

В данном примере не используются блоки и схемы, вычисления производятся непосредственно в теле проекта.

При комментировании кода остановимся на:
- операции сравнения – строки с 14 по 20;
- переходе на метку – строки с 22 по 25;
- операции присвоения значения – строки с 27 по 29.

Описание программы представлено в таблице 3.2

Т а б л и ц а 3.2

Переменные	idx : INTEGER, глобальная, индекс 0 Count : INTEGER, глобальная, индекс 1
Алгоритм	При каждом исполнении проекта (на каждом такте контроллера) переменная COUNT увеличивается на единицу до тех пор, пока значение IDX не достигло 10. Как только значение IDX достигло 10, цикл прекращается, переменная IDX обнуляется и исполнение проекта прекращается. Данная программа реализует цикл с пост-проверкой числа итераций, цикл с пред-проверкой числа итераций приведен в следующем примере.

Код: точка входа на метке L2	00: (*#################### Begin of << Project2 >> *) 01: (*.............................. "+" *) 02: L2: ILoad regA, Vars[1] 03: IMov regB, 1 04: IAdd regA, regB 05: IStore Vars[2], regA 06: IStore Vars[1], regA 07: (*.............................. "+" *) 08: L8: ILoad regA, Vars[0] 09: IMov regB, 1 10: IAdd regA, regB 11: IStore Vars[3], regA 12: IStore Vars[0], regA 13: (*.............................. "<" *) 14: L14: BMov regC, False 15: ILoad regA, Vars[0] 16: IMov regB, 10 17: ICmp regA, regB 18: JMP_GE L20 19: BMov regC, True 20: L20: BStore Vars[4], regC 21: (*.............................. "Условный GOTO" *) 22: L22: BLoad regA, Vars[4] 23: BMov regB, True 24: BCmp regA, regB 25: JMP_EQ L2 26: (*.............................. ":=" *) 27: L27: IMov regA, 0 28: IStore Vars[5], regA 29: IStore Vars[0], regA 00: L30: Ret 31: (********************** End of << Project2 >> *)
Сравнение: IDX < 10	
Строка 14	L14: BMov regC, False поместить в регистр C результат операции сравнения – по умолчанию считается, что результат равен FALSE (т.е. условие не выполнилось).
Строка 15	ILoad regA, Vars[0] загружаем в регистр A первый операнд – переменную IDX
Строка 16	IMov regB, 10 поместить в регистр B второй операнд – значение 10
Строка 17	ICMP regA, regB произвести операцию сравнения целых чисел. Если условие сравнения выполнилось, то флаги результатов будут иметь следующие значения ZF = 0, CF = 1.
Строка 18	L18: JMP_GE L20 если условие не выполнилось, то перейти на метку L20, которой помечена операция сохранения результата сравнения. Для определения невыполнения операции используется переход с обратным по отношению к заданному условием. В

	данном случае задано условие «меньше», следовательно, обратным к нему будет условие «больше или равно».
Строка 19	BMov regC, True поместить в регистр C значение TRUE, которое соответствует истинности сравнения.
Строка 20	L20: BStore Vars[4], regC сохранить результат сравнения «IDX < 10» в автоматическую переменную с индексом 4 (связана с выходом блока «<»).
Условный переход на метку Begin_Loop	
Строка 22	L22: BLoad regA, Vars[4] загрузить в регистр A результат сравнения IDX < 10.
Строка 23	BMov regB, True поместить в регистр B TRUE в качестве значения, с которым должен быть сравнен результат сравнения IDX < 10
Строка 24	BCMP regA, regB сравнить два регистра, т.е. проверить: на вход команды перехода подано TRUE или FALSE.
Строка 25	L25: JMP_EQ L2 если подано TRUE (ZF = 1, CF = 0), то перейти на метку Loop_Begin (в теле кода метка L2), иначе – продолжить исполнение.
Операция присвоения: IDX := 0	
Строка 27	L27: IMov regA, 0 поместить в регистр A значение, которое должно быть присвоено переменной IDX.
Строка 28	IStore Vars[5], regA поместить это значение в автоматическую переменную с индексом 5, связанную с выходом блока «:=» (явно лишняя операция).
Строка 29	IStore Vars[0], regA поместить значение 0 в переменную IDX.

3 ПРИМЕР 3: ОРГАНИЗАЦИЯ ЦИКЛА С ПРЕД-ПРОВЕРКОЙ ЧИСЛА ИТЕРАЦИЙ

Программа на языке FBD представлена на рисунке 3.3. Номера над блоками означают порядок исполнения.

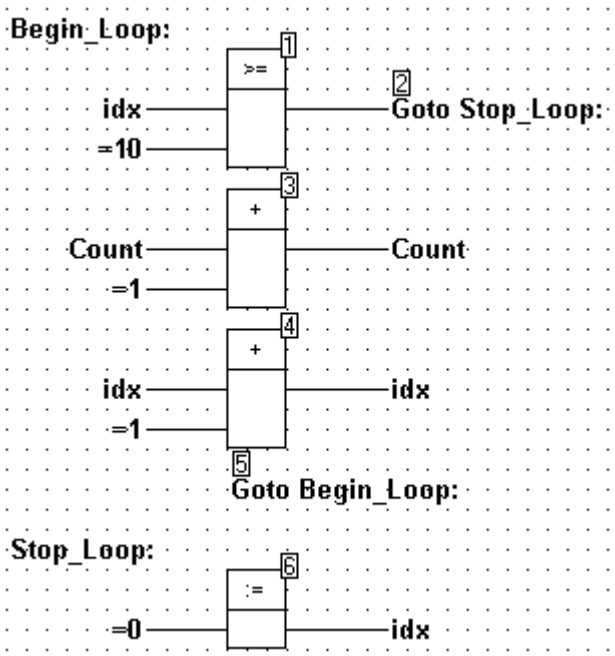


Рисунок 3.3

В данном примере не используются блоки и схемы, вычисления производятся непосредственно в теле проекта.

Подробно комментировать данную программу не имеет смысла, отметим лишь отличия от предыдущего примера:

- проверка необходимости завершения и выход из цикла производится парой элементов 1 и 2;
- собственно цикл организован при помощи элемента 5 (безусловный переход)
- данная реализация цикла несколько длиннее предыдущей (дополнительно использован безусловный переход), но более проста при прочтении.

Код, полученный при компиляции данной программы, приведен ниже.

```
(********************************** Begin of << Project2 >> *)
(*.................................. ">=" *)
L2:      BMov regC, False
         ILoad regA, Vars[0]
         IMov regB, 10
         ICmp regA, regB
         JMP_LT L8
         BMov regC, True
L8:      BStore Vars[2], regC
(*................................. "Условный GOTO" *)
L10:     BLoad regA, Vars[2]
         BMov regB, True
         BCmp regA, regB
```

```
         JMP_EQ L29
(*................................ "+" *)
L15:     ILoad regA, Vars[1]
         IMov regB, 1
         IAdd regA, regB
         IStore Vars[3], regA
         IStore Vars[1], regA
(*................................ "+" *)
L21:     ILoad regA, Vars[0]
         IMov regB, 1
         IAdd regA, regB
         IStore Vars[4], regA
         IStore Vars[0], regA
(*............................... "Безусловный GOTO" *)
L27:     JMP L2
(*................................ ":=" *)
L29:     IMov regA, 0
         IStore Vars[5], regA
         IStore Vars[0], regA
L32:     Ret
(********************************* End of << Project2 >> *)
```

4 ПРИМЕР 4: УПРАВЛЕНИЕ TIME-ПЕРЕМЕННОЙ

Программа на языке FBD представлена на рисунке 3.4. Номера над блоками означают порядок исполнения.

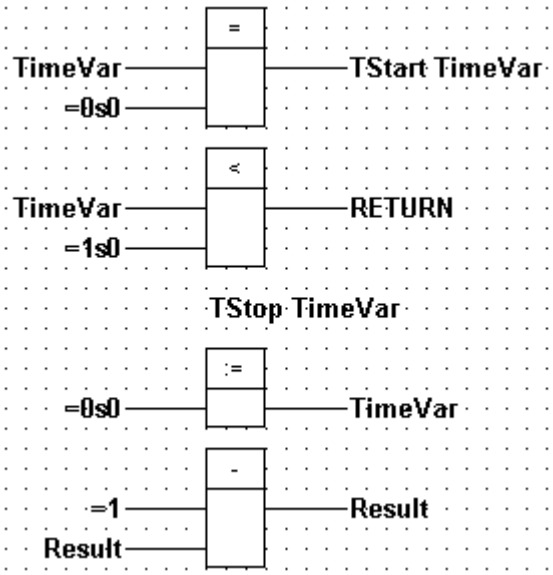


Рисунок 3.4

Диаграмма работы представлена на рисунке 3.5.

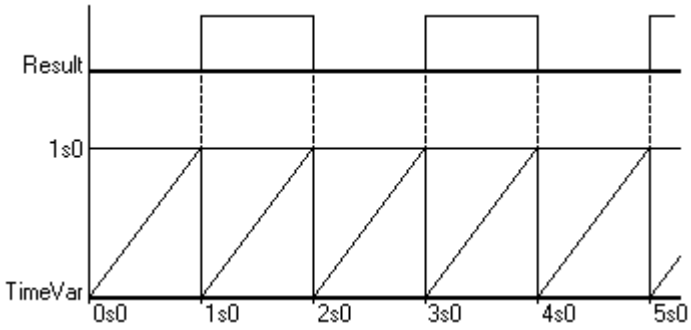


Рисунок 3.5

В данном примере не используются блоки и схемы, вычисления производятся непосредственно в теле проекта.

Подробно разберем следующие участки кода:
- условное управление TIME-переменной: строки с 11 по 14;
- условный возврат: строки с 24 по 27;
- безусловное управление TIME-переменной: строка 28.

Описание программы представлено в таблице 3.3

Т а б л и ца 3.3

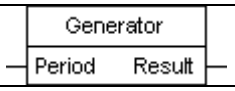
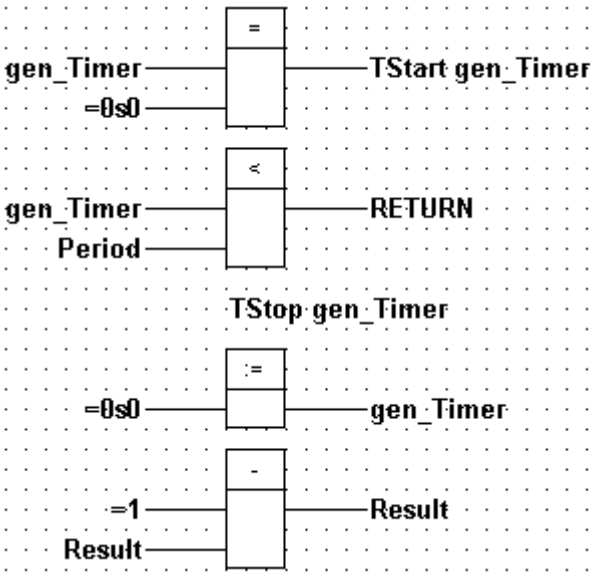
Переменные	Result : INTEGER, глобальная, индекс 0 TimeVar : TIME, глобальная, индекс 1
Алгоритм	Если TimeVar = 0s0, то для нее производится запуск накопления времени. Затем, если значение TimeVar меньше 1s0, то исполнение прерывается, иначе накопление времени останавливается, значение TimeVar обнуляется, а выход (Result) меняется с 0 на 1 или наоборот. Таким образом, представленная программа не что иное, как генератор прямоугольных импульсов.
Код: точка входа на метке L2	01: (*########################## Begin of << Project2 >> *) 02: (*.................................. "=" *) 03: L2: BMov regC, False 04: TmLoad regA, Vars[1] 05: TmMov regB, 0s0 06: TmCmp regA, regB 07: JMP_NE L8 08: BMov regC, True 09: L8: BStore Vars[2], regC 10: (*..................... "Условный TIMER_CONTROL" *) 11: L10: BLoad regA, Vars[2] 12: BMov regB, True 13: BCmp regA, regB 14: TStart_EQ Vars[1] 15: (*.................................. "<" *) 16: L15: BMov regC, False 17: TmLoad regA, Vars[1] 18: TmMov regB, 1s0 19: TmCmp regA, regB 20: JMP_GE L21 21: BMov regC, True 22: L21: BStore Vars[3], regC 23: (*.............................. "Условный RETURN" *) 24: L23: BLoad regA, Vars[3] 25: BMov regB, True 26: BCmp regA, regB 27: Ret_EQ 28: L27: TStop Vars[1] 29: (*.................................. ":=" *) 30: L29: TmMov regA, 0s0 31: TmStore Vars[4], regA 32: TmStore Vars[1], regA 33: (*.................................. "-" *) 34: L33: IMov regA, 1 35: ILoad regB, Vars[0] 36: ISub regA, regB 37: IStore Vars[5], regA 38: IStore Vars[0], regA 39: L38: Ret 40: (*************************** End of << Project2 >> *)
Условное управление таймерной переменной	
Строка 11	L10: BLoad regA, Vars[2] загрузить в регистр А результат сравнения TimeVar = 0s0
Строка 12	BMov regB, True поместить в регистр В TRUE в качестве значения, с которым должен

	быть сравнен результат сравнения TimeVar = 0s0
Строка 13	BCMP regA, regB сравнить два регистра, т.е. проверить: на вход команды управления TIME-переменной подано TRUE или FALSE.
Строка 14	TStart_EQ Vars[1] если подано TRUE (ZF = 1, CF = 0), то запустить накопление значения в переменной TimeVar, иначе – продолжить исполнение.
Условный возврат	
Строка 24	L23: BLoad regA, Vars[3] загрузить в регистр A результат сравнения TimeVar < 1s0
Строка 25	BMov regB, True поместить в регистр B TRUE в качестве значения, с которым должен быть сравнен результат сравнения TimeVar < 1s0
Строка 26	BCMP regA, regB сравнить два регистра, т.е. проверить: на вход команды возврата подано TRUE или FALSE.
Строка 27	Ret_EQ если подано TRUE (ZF = 1, CF = 0), то завершить исполнение проекта, иначе – продолжить исполнение.
Безусловное управление таймерной переменной	
Строка 28	L27: TStop Vars[1] остановить накопление значения переменной TimeVar

5 ПРИМЕР 5: ИСПОЛЬЗОВАНИЕ БЛОКА ВЫЧИСЛЕНИЙ

В данном примере продемонстрировано использование блоков вычислений. Программа из предыдущего примера оформлена в виде блока под именем Generator. Описание блока приведено в таблице 3.4.

Т а б л и ц а 3.4

Внешний вид	
Входы	PERIOD : TIME
Выход	RESULT : INTEGER
Переменные	GEN_TIMER : TIME
Программа на FBD	
Описание	Программа повторяет предыдущий пример, за исключением: • вместо глобальной переменной TimeVar используется локальная переменная блока gen_Timer; • при определении длины полупериода используется не константа, а входная переменная Period.

Код проекта на языке FBD представлен на рисунке 3.6.

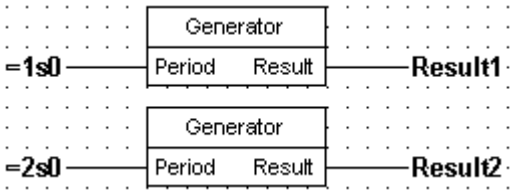


Рисунок 3.6

Диаграмма работы для первого экземпляра блока Generator представлена на рисунке 3.7.

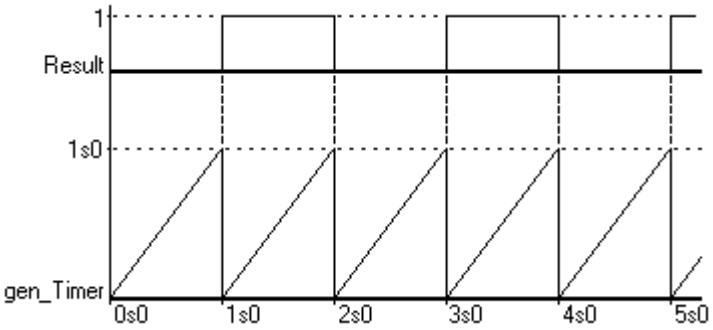


Рисунок 3.7

Диаграмма работы для второго экземпляра блока Generator представлена на рисунке 3.8.

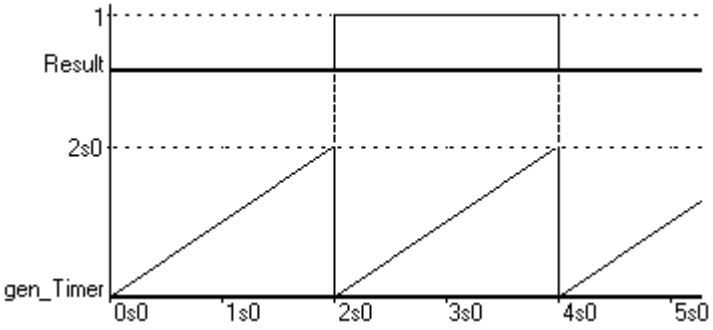


Рисунок 3.8

При комментировании кода остановимся на вызове блока GENERATOR (строки с 44 по 52).

Примечание: Обратите внимание на то, что код блока GENERATOR (строки с 3 по 39) оперирует с индексами переменных, которые начинаются с 10000. Это говорит о том, что используются локальные переменные блока.

Описание программы приведено в таблице 3.5.

Т а б л и ц а 3.5

Переменные	Result1 : INTEGER, глобальная, индекс 0 Result2 : INTEGER, глобальная, индекс 1
Алгоритм	В программе используются два генератора прямоугольных импульсов, реализованных в виде блока GENERATOR. В первом случае полупериод сигнала равен 1s0, во втором – 2s0 (т.е. 1 и 2 секунды соответственно).
Код: точка входа на метке L2	01: (*########################## Begin of << Generator >> *) 02: (*.................................. "=" *) 03: L2: BMov regC, False 04: TmLoad regA, Vars[10002] 05: TmMov regB, 0s0 06: TmCmp regA, regB 07: JMP_NE L8 08: BMov regC, True 09: L8: BStore Vars[10003], regC 10: (*.......................... "Условный TIMER_CONTROL" *) 11: L10: BLoad regA, Vars[10003] 12: BMov regB, True

```
13:          BCmp regA, regB
14:          TStart_EQ Vars[10002]
15: (*.................................. "<" *)
16: L15:     BMov regC, False
17:          TmLoad regA, Vars[10002]
18:          TmLoad regB, Vars[10000]
19:          TmCmp regA, regB
20:          JMP_GE L21
21:          BMov regC, True
22: L21:     BStore Vars[10004], regC
23: (*.................................. "Условный RETURN" *)
24: L23:     BLoad regA, Vars[10004]
25:          BMov regB, True
26:          BCmp regA, regB
27:          Ret_EQ
28: L27:     TStop Vars[10002]
29: (*.................................. ":=" *)
30: L29:     TmMov regA, 0s0
31:          TmStore Vars[10005], regA
32:          TmStore Vars[10002], regA
33: (*.................................. "-" *)
34: L33:     IMov regA, 1
35:          ILoad regB, Vars[10001]
36:          ISub regA, regB
37:          IStore Vars[10006], regA
38:          IStore Vars[10001], regA
39: L38:     Ret
40: (****************************** End of << Generator >> * )
41:
42: (*########################### Begin of << Project2 >> *)
43: (*............................... "InVars : Generator" *)
44: L43:     TmMov regA, 1s0
45:          Set_DS GDS[5]
46:          TmStore Vars[10000], regA
47: L46:     Call L2
48: (*............................... "OutVars : Generator" *)
49:          ILoad regA, Vars[10001]
50:          Set_DS LDS[10007]
51:          IStore Vars[2], regA
52:          IStore Vars[0], regA
53: (*............................... "InVars : Generator" *)
54:          TmMov regA, 2s0
55:          Set_DS GDS[6]
56:          TmStore Vars[10000], regA
57: L56:     Call L2
58: (*............................... "OutVars : Generator" *)
59:          ILoad regA, Vars[10001]
60:          Set_DS LDS[10007]
61:          IStore Vars[3], regA
62:          IStore Vars[1], regA
63: L62:     Ret
64: (****************************** End of << Project2 >> *)
```

Вызов блока GENERATOR

Строка 44	L43: TmMov regA, 1s0 значение, подаваемое на входную переменную блока Period помещаем в регистр А. В данном примере у блока только один входной параметр, поэтому использован только регистр А, если параметров больше, то значение второго параметра будет помещено в регистр В, третьего – в регистр С и т.д.

Строка 45	SetDS GDS[5] указываем, что текущим локальным блоком переменных будет блок с индексом, хранящимся в глобальной переменной с индексом 5 (GDS = Global DataSegment). Именно этот блок переменных хранит значения переменных для данного экземпляра вычисляемого блока GENERATOR.
Строка 46	TmStore Vars[10000], regA помещаем значение из регистра A, которое должно поступить на первый вход блока во входную переменную блока PERIOD (ведь параметр PERIOD является первым входным параметром).
Строка 47	L46: Call L2 а теперь с чистой совестью вызываем код блока GENERATOR – пусть его исполняется.
Строка 49	ILoad regA, Vars[10001] код блока как бы исполнился и теперь надо разложить результаты вычислений, т.е. значения выходных переменных блока по местам. Для этого значения выходных переменных блока записываем в регистры: в регистр A – первая выходная переменная, в регистр B – вторая и т.д.
Строка 50	Set_DS LDS[10007] восстанавливаем тот блок локальных переменных, который использовался до вызова. В нашем случае индекс блока хранится в локальной переменной только что исполнявшегося блока, индекс переменной 10007.
Строка 51	IStore Vars[2], regA теперь положим значение выхода блока в автоматическую переменную, связанную с выходом блока (в данном случае бесполезная операция – где же оптимизатор?).
Строка 52	IStore Vars[0], regA И, наконец, положим значение выхода блока в переменную RESULT1, ведь именно она, судя по программе должна хранить значение, полученное при исполнении блока GENERATOR.

Вызов второго экземпляра блока Generator отличается только значением входного параметра.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					
1				все	17				