**Язык списка инструкций il**

Текстовый язык IL (Instruction List) относится к ассемблероподобным языкам, т. е. к языкам низкого (машинного) уровня, что существенно облегчает, например, вызов функциональных блоков и функций «условно» или «безусловно», выполнение назначений и условных или безусловных переходов внутри секции.

Язык IL, позволяет создавать высокоэффективные и оптимизи­рованные функции. Его можно рекомендовать для написания наи­более критических мест в программе.

Ассемблероподобные языки использовались для программиро­вания компьютеров в 50-е годы прошлого века и все еще предла­гаются некоторыми изготовителями ПЛК, особенно программи­стами, поддерживающими микроРС. Программу можно написать с помощью любого текстового редактора.

При вводе ключевых слов, разделителей и комментариев в ре­дакторе предусмотрена непосредственная по буквам проверка. При обнаружении ключевого слова, разделителя или комментария они идентифицируются через цветовое оттенение. При введении не­санкционированных ключевых слов (назначений или операторов) они будут также идентифицироваться через цветовое оттенение.

Согласно стандарту IEC 61131-3 ключевые слова должны быть введены в символах верхнего регистра.

Пробелы и метки табуляции не влияют на синтаксис, они мо­гут использоваться везде.

Генерация объектного кода вместе с проверкой синтаксиса бу­дет выполняться, когда секция закрыта. Любые найденные ошибки затем отображаются в окне сообщений. Однако проверка синтак­сиса может также осуществляться во время создания программы или блока DFB командой Project → Analyze Section.

**Команды (инструкции) языка IL**

Список инструкций составлен из последовательности команд. Каждая команда начинается в новой ячейке и состоит из операто­ра (в случае необходимости с модификатором) и (если требуется для соответствующей операции) сопровождается одним или несколькими операндами (рисунок 9.1). Если используется несколько операндов, они отделяются запятыми. Команде может предшествовать метка с двоеточием. Команда может сопровождаться комментарием.

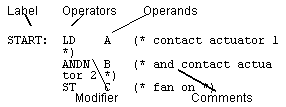


Рисунок 9.1 – Формат записи команды в языке IL

Язык IL – так называемый аккумуляторно-ориентированный язык, т. е. каждая команда использует или изменяет текущее содержимое аккумулятора (тип временной памяти). Стандарт IEC 61131-3 обозначает этот аккумулятор как результат.

Список команд должен всегда начинаться с оператора LD (ко­манда загрузки аккумулятора) и заканчиваться оператором сохра­нения ST.

Пример сложения:

LD 10

ADD 25

ST A

Пример показывает загрузку литерала 10 в аккумулятор, добавление литерала 25 и внесение результата в переменную А. Содержимое переменной и аккумулятора теперь 35. Любая после­дующая команда работала бы с содержимым аккумулятора 35, если она не начинается с LD.

Операции сравнения также всегда касаются аккумулятора. Булев результат сравнения вносится в аккумулятор, следовательно, это является текущим содержанием аккумулятора.

Пример сравнения:

LD B

GT 10

В примере значение переменной В загружено в аккумулятор и сравнивается с литералом 10. Если В меньше или равно 10, содержимое аккумулятора равно 0 (FALSE). Если В больше чем 10, содержимое аккумулятора есть 1 (TRUE).

**Операнды.** Операндом может быть литерал, переменная, структурированная переменная, элемент структурированной переменной, выход FB/DFB или прямой адрес.

Когда осуществляется доступ к массивам переменных (ARRAY), элемент указателя допускает только литералы и переменные типа ANY\_INT.

Пример сохранения:

LD var1[i]

ST var2.otto[4]

В таблица 9.1 приведены заданные по умолчанию типы данных прямого адреса.

Таблица 9.1

|  |  |  |  |
| --- | --- | --- | --- |
| Вход | Выход | Заданные по умолчанию типы данных | Возможные типы данных |
| %IХ,%I | %QX,%Q | BOOL | BOOL |
| %IВ | %QB | BYTE | BYTE |
| %IW | %QW | INT | INT, UINT, WORD |
| %ID | %QD | REAL | REAL, DINT, UDINT, TIME |

Назначение типов данных, отличных от заданных по умолчанию типов данных прямого адреса, должно выполняться через явное объявление (VAR...END\_VAR). В пакете Concept объявление VAR...END\_VAR не может использоваться для объявления переменных. Объявление переменных очень легко выполнить через редактор переменных.

Операнд и текущее содержимое аккумулятора должны иметь одинаковый тип данных. Если необходимо обработать операнды различных типов данных, сначала выполняется преобразование типов. Исключением является тип данных TIME вместе с арифметическими операторами MUL и DIV. Эти два оператора позволяют обработать операнд типа данных TIME вместе с операндом типа данных ANY\_NUM. В этом случае результат этих команд будет иметь тип данных TIME.

Пример преобразования:

LD i1

INT\_TO\_REAL

ADD r4

ST r3

В примере целая переменная i1 преобразована в вещественную переменную прежде, чем она добавлена к вещественной перемен­ной r4.

Пример умножения:

LD t2

MUL i4

ST t1

В примере переменная времени t2 умножена на целую пере­менную i4, а результат сохраняется в переменной времени t1.

**Модификаторы.** Модификаторы влияют на выполнение пред­шествующего оператора.

Модификатор N используется, чтобы инвертировать побитно значение операнда. Модификатор N может применяться только к операндам типа данных ANY\_BIT.

Пример модификатора N:

LD A

ANDN B

ST C

В примере С = 1, если А = 1 и В = 0.

Модификатор С используется, чтобы выполнить соответствующую команду, если значение аккумулятора равно 1 (TRUE). Модификатор С может применяться только к операндам типа данных BOOL.

Пример модификатора С:

LD A

AND B

JMPC START

В примере переход к START выполняется, только если А = 1 (TRUE) и В = 1 (TRUE).

Если модификатор С объединен с модификатором N, соответствующая команда выполняется, только если значение аккумуля­тора равно булеву 0 (FALSE).

Пример CN:

LD A

AND B

JMPCN START

В примере переход к START выполняется, только если А = 0(FALSE) и/или В = 0 (FALSE).

Модификатор **(** (левая круглая скобка) используется, чтобы задержать оценку операнда до появления оператора (правая круглая скобка). Число операций правой круглой скобки должно быть рав­ным числу модификаторов левой круглой скобки. Круглые скобки могут быть вложенными.

Пример:

LD A

AND B

AND(C

OR D

)

ST E

В примере Е будет равно 1 если С и/или D равны 1, а А и B равны 1.

Этот же пример может также программироваться следующим образом:

LD A

AND В

AND(

LD C

OR D

)

ST E

**Операторы.**

Оператор является символом для арифметической или логической операции, которая будет выполнена, или для вызова функции.

Операторы являются обобщенными, т. е. они автоматически корректируются к типу данных операнда.

Операторы языка программирования IL приведены в таблицах 9.2-9.8.

Таблица 9.2 – Операторы загрузки и сохранения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| LD | N  (только для операндов типа BOOL, BYTE, WORD или DWORD) | Загрузка значений операндов в аккумулятор | Литерал, переменная, прямой адрес любого типа данных | Значение операнда загружается в аккумулятор, используя LD. Размер данных аккумулятора автоматически подстраивается для типа данных операции. Это также действительно для производных типов данных.  Пример: В примере значение A загружается в аккумулятор с добавлением значения B , и результат сохраняется в E.  LD A  ADD B  ST E |
| ST | N  (только для операндов типа данных BOOL, BYTE, WORD или DWORD) | Сохранение значения аккумулятора в операнде | переменная, прямой адрес любого типа данных | Текущее значение аккумулятора сохраняется в операнде, используя ST. тип данных операнда должен соответствовать "типу данных" аккумулятора.  Пример: В этом примере значение A загружается в аккумулятор с добавлением значения B , и результат сохраняется в E.  LD A  ADD B  ST E  "Старый" результат используется далее, в зависимости от того, следует ли LD за ST или нет.  Пример: В этом примере значение A загружается в аккумулятор с добавлением значения B , и результат сохраняется в E. Значение E (текущее содержимое аккумулятора) вычитается из значения B , и результат сохраняется в C.  LD A  ADD B  ST E  SUB 3  ST C |

Таблица 9.3 – Операторы установки и сброса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| S | - | Установка операнда в 1, когда содержимое аккумулятора = 1 | Переменная, прямой адрес данных типа BOOL | S установка операнда в "1", когда текущее значение аккумулятора равно логической 1.  Пример: В этом примере значение A загружается в аккумулятор, если содержимое аккумулятора (значение A) = 1, тогда OUT устанавливается в 1.  LD A  S OUT  Обычно этот оператор используется вместе с оператором сброса R в паре.  Пример: В этом примере показан RS Триггер (главный сброс), который управляется через логические переменные A и C.  LD A  S OUT  LD C  R OUT |
| R | - | Установка операнда в 0, когда содержимое аккумулятора = 1 | Переменная, прямой адрес данных типа BOOL | R установка операнда в "0", когда текущее содержимое аккумулятора равно логической 1.  Пример: В этом примере значение A загружается в аккумулятор, если содержимое аккумулятора (значение A) = 0, тогда OUT устанавливается 1.  LD A  R OUT  Обычно этот оператор используется вместе с оператором установки S в паре.  Пример: В этом примере показан SR Триггер (главная установка), который управляется через две логические переменные A и C.  LD A  R OUT  LD C  S OUT |

Таблица 9.4 – Логические операторы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| AND | N, N(, ( | Логическое И | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD или DWORD | При помощи AND производится логическая операция И между содержимым аккумулятора и операндом.  Для BYTE, WORD и DWORD типов данных операция производится побитно.  Пример: В примере D = 1, если A, B и C = 1.  LD A  AND B  AND C  ST D |
| OR | N, N(, ( | Логическое ИЛИ | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD или DWORD | При помощи OR производится логическая операция ИЛИ между содержимым аккумулятора и операндом.  Для BYTE, WORD и DWORD типов данных, связь производится побитно.  Пример: В примере D = 1, если A или B = 1 и C = 1.  LD A  OR B  OR C  ST D |
| XOR | N, N(, ( | Логическое исключающее ИЛИ | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD или DWORD | При помощи XOR производится логическая операция Исключающее ИЛИ между содержимым аккумулятора и операндом.  Если связывается более двух операндов, результат для нечетного числа 1-состояния = 1, и = 0 для четного числа единичных состояний.  Для BYTE, WORD и DWORD типов данных, связь производится побитно.  Пример: В примере D = 1, если A или B = 1. Если A и B имеют одинаковое состояние (оба 0 или 1), D = 0.  LD A  XOR B  ST D  Если связывается более двух операндов, результат для нечетного числа единичных состояний = 1, и 0 для четного числа единичных состояний.  Пример: В примере F = 1, если 1 или 3 операнды = 1. F = 0, если 0, 2 или 4 операнды = 1.  LD A  XOR B  XOR C  XOR D  XOR E  ST F |
| NOT | - | Логическое отрицание | Содержимое аккумулятора типов данных BOOL, BYTE, WORD или DWORD | Содержимое аккумулятора инвертируется NOT.  Пример: В примере B = 1, если A = 0 и B = 0, если A = 1.  LD A  NOT  ST B |

Таблица 9.5 – Арифметические операторы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| ADD | ( | Сложение | Литерал, переменная, прямой адрес данных типа INT, DINT, UINT, UDINT, REAL или TIME | При помощи ADD значение операнда прибавляется к значению аккумулятора.  Пример: Пример соответствует формуле D = A + B + C  LD A  ADD B  ADD C  ST D |
| SUB | ( | Вычитание | Литерал, переменная, прямой адрес данных типа INT, DINT, UINT, UDINT, REAL или TIME | При помощи SUB значение операнда вычитается из содержимого аккумулятора.  Пример: Пример соответствует формуле D = A - B - C  LD A  SUB B  SUB C  ST D |
| MUL | ( | Умножение | Литерал, переменная, прямой адрес данных типа INT, DINT, UINT, UDINT, REAL или TIME | При помощи MUL содержимое аккумулятора умножается на значение оператора.  Пример: Пример соответствует формуле D = A \* B \* C  LD A  MUL B  MUL C  ST D |
| DIV | ( | Деление | Литерал, переменная, прямой адрес данных типа INT, DINT, UINT, UDINT, REAL или TIME | При помощи DIV содержимое аккумулятора делится на значение операнда.  Пример: Пример соответствует формуле D = A / B / C  LD A  DIV B  DIV C  ST D |
| MOD | ( | Деление по модулю | Литерал, переменная, прямой адрес данных типа INT, DINT, UINT, UDINT, REAL или TIME | Для MOD значение первого операнда делится на значение второго операнда, и остаток от деления (модуль) возвращается как результат.  Пример: В примере  C = 1, если A = 7 и B = 2  C = 1, если A = 7 и B = -2  C = -1, если A = -7 и B = 2  C = -1, если A = -7 и B = -2  LD A  MOD B  ST C |

Таблица 9.6 – Операторы сравнения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| GT | ( | Сравнение: > | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи GT содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора больше, чем содержимое операнда, результат – логическая 1. Если содержимое аккумулятора меньше или равно содержимому операнда, результат – логический 0.  Пример: В примере значение D = 1, если A больше 10, иначе значение D = 0.  LD A  GT 10  ST D |
| GE | ( | Сравнение: >= | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи GE содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора больше или равно содержимому операнда, результат – логическая 1. Если содержимое аккумулятора меньше содержимого операнда, результат – логический 0.  Пример: В примере значение D = 1, если A больше или равно 10, иначе, значение D = 0.  LD A  GE 10  ST D |
| EQ | ( | Сравнение: = | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи EQ содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора равно содержимому операнда, результат – логическая 1. Если содержимое аккумулятора не равно содержимому операнда, результат – логический 0.  Пример: В примере значение D = 1, если A равно 10, иначе значение D = 0.  LD A  EQ 10  ST D |
| NE | ( | Сравнение: <> | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи NE содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора не равно содержимому операнда, результат – логическая 1. Если содержимое аккумулятора равно содержимому операнда, результат – логический 0.  Пример: В примере значение D = 1, если A не равно 10, иначе значение D = 0.  LD A  NE 10  ST D |
| LE | ( | Сравнение: <= | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи LE содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора меньше или равно содержимому операнда, результат – логическая 1. Если содержимое аккумулятора больше содержимого операнда, результат – логический 0.  Пример: В примере значение D = 1, если A меньше или равно 10, иначе значение D = 0.  LD A  LE 10  ST D |
| LT | ( | Сравнение: < | Литерал, переменная, прямой адрес данных типа BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT или TOD | При помощи LT содержимое аккумулятора сравнивается с содержимым операнда. Если содержимое аккумулятора меньше, чем содержимое операнда, результат – логическая 1. Если содержимое аккумулятора больше или равно содержимому операнда, результат – логический 0.  Пример: В примере значение D = 1, если A меньше 10, иначе значение D = 0.  LD A  LT 10  ST D |

Таблица 9.7 – Операторы вызова

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| CAL | C, CN  (только если содержимое аккумулятора BOOL типа данных) | Вызов функционального блока, DFB или подпрограммы | Имя экземпляра функционального блока, DFB или подпрограммы | Функциональный блок, DFB или подпрограмма вызывается в ограниченном или неограниченном виде CAL. |
| FUNCTIONSNAME | - | Выполнение функции | Литерал, переменная, прямой адрес (тип данных зависит от функции) | Функция выполняется по имени функции. |
| PROCEDURENAME | - | Выполнение процедуры | Литерал, переменная, прямой адрес (тип данных зависит от процедуры) | Процедура выполняется по имени процедуры. |

Таблица 9.8 – Операторы структурирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Модификатор** | **Описание** | **Операнды** | **Описание** |
| JMP | C, CN  (только если содержимое аккумулятора BOOL типа данных) | Прыжок к метке | TAG | При помощи JMP прыжок к метке может быть ограниченным и неограниченным. |
| RET | C, CN  (только если содержимое аккумулятора BOOL типов данных) | Возврат к следующей более высокой организационной единице программы | - | Каждая подпрограмма и каждый DFB (производный функциональный блок) покидается после выполнения, т.е. происходит возврат к основной вызывающей программе.  Если подпрограмма/DFB покидается преждевременно, возврат к основной программе может форсироваться, используя RET (Возврат).  RET может использоваться только в подпрограммах или DFBs. Они не могут использоваться в основной программе. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ) | - | редактирование отложенных операций | - | При помощи закрывающей скобки ) начинается выполнение восстановленного оператора. Число закрывающих скобок должно быть равно числу открывающих скобок. Скобки могут быть вложенными.  Пример: В примере E = 1, если C и/или D = 1 и A и B = 1.  LD A  AND B  AND( C  OR D  )  ST E |

**Метки и прыжки**

Метки служат целями для прыжков.

Свойства меток:

* Метки всегда должны быть первыми элементами в строке.
* Имя должно быть свободным в пределах директории и не является чувствительным к регистру.
* Метки могут быть 32 символа длиной (максимум).
* Метки должны соответствовать IEC соглашениям об именовании.
* Метки отделяются двоеточием : от следующей инструкции.
* Метки разрешены только в начале"Выражений", иначе в батарее может быть обнаружено неопределенное значение.

Пример:

start: LD A

AND B

OR C

ST D

JMP start

Свойства прыжков:

* При помощи JMP прыжок к метке может быть ограниченным или неограниченным.
* JMP может использоваться с модификаторами C и CN (только если содержимое батарей типа данных BOOL).
* Прыжки могут производится в пределах программы и секций DFB.
* Прыжки возможны только в текущей секции.

Возможные места назначения:

* первая LD инструкция вызова EFB/DFB с присваиванием входных параметров (см. start2),
* нормальная LD инструкция (см. start1),
* CAL инструкция, которая не работает с присваиванием входных параметров (см. start3),
* JMP инструкция (см. start4),
* конец списка инструкций (см. start5).

Пример

start2: LD A

ST counter.CU

LD B

ST counter.R

LD C

ST counter.PV

CAL counter

JMPCN start4

start1: LD A

AND B

OR C

ST D

JMPC start3

LD A

ADD E

JMP start5

start3: CAL counter (

CU:=A

R:=B

PV:=C )

JMP start1

start4: JMPC start1

start5:

Команда VAR предназначена для объявления используемых функциональных блоков и для объявления прямых адресов, если они не должны использоваться с заданным по умолчанию типом данных. В пакете Concept команда VAR не может применяться для объявления переменных. Объявление переменных очень легко выполняется через редактор переменных.

Команда END\_VAR идентифицирует конец объявления.

Команда VAR...END\_VAR вводится только один раз в начале секции. Все блоки FB/DFB и используемые прямые адреса, кото­рые отличаются от заданного по умолчанию типа данных, должны быть здесь объявлены.

Объявление блоков FB/DFB и прямых адресов применяется только к текущей секции. Если тот же самый тип FFB или тот же самый адрес должен также использоваться в другой секции, тип FFВ или адрес соответственно должен быть объявлен снова в этой секции.

**Объявление функциональных блоков и блоков DFB.** Перед вызовом функционального блока или блока DFB он должен быть объявлен с использованием команд VAR и END\_VAR. В этом объявлении каждому применяемому экземпляру блока FB/DFB назначается имя. Имя экземпляра должно быть уникально для всего проекта; нет никакого различия между заглавными или строч­ными буквами. Имя экземпляра должно удовлетворять соглашениям стандарта по имени в противном случае появится сообщение об ошибках.

Тип функционального блока, например CTD\_DINT, ROL\_WORD, SIN\_REAL, будет введен после имени экземпляра.

Для обобщенных типов функциональных блоков (например, MUX, SEL) не имеется никакого обозначения типа данных. Оно будет определено типом данных фактических параметров. Если фактические параметры состоят из литералов, для функционального блока будет принят тип данных INT.

Любое число имен экземпляра может быть объявлено для бло­ков FB/DFB.

Пример объявления функциональных блоков и DFB приведен на рисунке 9.2.

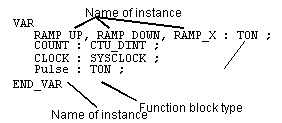


Рисунок 9.2 – Объявление функциональных блоков и DFB

**Объявление прямых адресов.** В этом объявлении каждому используемому прямому адресу, который имеет тип данных, отличающийся от заданного по умолчанию типа данных, назначается тип данных по выбору.

Пример объявление прямых адресов приведен на рисунке 9.3.

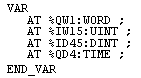


Рисунок 9.3 – Объявление прямых адресов

**Вызовы функциональных блоков и блоков DFB.** Чтобы найти имена доступных функциональных блоков и блоков DFB, нужно обратиться к библиотекам блоков. Существуют три способа для вызова функционального блока и блока DFB:

* оператором с CAL и списком входных параметров;
* оператором CAL и загрузкой или сохранением входных параметров;
* с помощью входных операторов.

Если функциональный блок не имеет входов или входы не будут параметрированы, функциональный блок должен все равно вызываться прежде, чем его выходы могут использоваться.

Каждый экземпляр FB/DFB может вызываться только один раз.

**Оператор CAL со списком входных параметров.** Функциональные блоки и DFB могут вызываться командой, состоящей из команды CAL, следующей за именем экземпляра FB/DFB и списка распределения значений (фактических параметров) к формальным параметрам в круглых скобках. Порядок распечатки формальных параметров в обращении функционального блока не имеет значения. Список фактических параметров может иметь конец строки сразу же после запятой. Не требует­ся, чтобы всем формальным параметрам были присвоены значе­ния. Если формальному параметру не задано значение, началь­ное значение, определенное в редакторе переменных, будет ис­пользоваться при выполнении функционального блока. Если начальное значение не было определено, будет использоваться значение по умолчанию (0).

Команда CAL (..) завершает назначение параметров функцио­нального блока и DFB. После этого передача значений в FB/DFB больше невозможна. Теперь могут считываться только выходные значения.

Примеры CAL со списком входных параметров приведены на рисунках 9.4 и 9.5

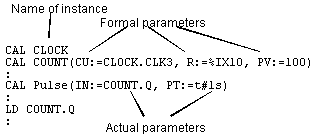


Рисунок 9.4 – Пример использования команды CAL (вариант 1)

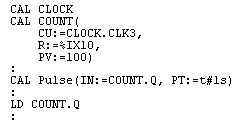


Рисунок 9.5 – Пример использования команды CAL (вариант 2)

**Оператор CAL с загрузкой и сохранением входных пара­метров.** Функциональные блоки и DFB могут вызываться списком команд, состоящим из команд загрузки фактических пара­метров, сохранения в формальных параметрах и вызова команды CAL. Порядок загрузки и сохранения параметров не имеет значе­ния. Не требуется, чтобы всем формальным параметрам были заданы значения. Если формальному параметру не присвоено значение, начальное значение, определенное в редакторе переменных, будет использоваться при выполнении функцио­нального блока. Если начальное значение не было определено, будет использоваться значение по умолчанию (0).

Команда CAL FBNAME завершает назначение параметров функциональных блоков и DFB. После этого передача значения этих параметров в блок FB/DFB больше невозможна. Теперь могут считываться лишь их выходные значения.

Только команды загрузки и сохранения для текущих назначений параметров FB/DFB позволяются между первой командой загрузки фактического параметра и вызовом функционального блока и блока DFB. Никакие другие команды не разрешаются в этом состоянии.

Примеры команды CAL с загрузкой и сохранением входных параметров приведены на рисунке 9.6.

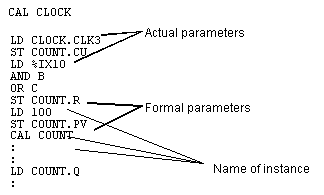


Рисунок 9.6 – Пример команды CAL с загрузкой и сохранением входных параметров

**Использование входных операторов.** Функциональные блоки могут вызываться списком команд, состоящим из команд загрузки фактических параметров, сохранения в формальны параметры, входного оператора. Порядок загрузки и сохранения параметров не имеет значения. Список фактических параметров может иметь конец строки сразу же после запятой. Не требуется, чтобы всем формальным параметрам были заданы значения. Если формальному параметру не задано значение, начальное значение, определенное в редакторе переменных, будет использоваться при выполнении функционального блока. Если начальное значение не было определено, будет использоваться значение по умолчанию (0).

Возможные входные операторы для различных функциональных блоков находятся по таблице 9.9. Никакие другие входные операторы не доступны.

Таблица 9.9 – Возможные входные операторы

|  |  |
| --- | --- |
| Входной оператор | Тип FB |
| S1, R | SR |
| S, R1 | RS |
| CLK | R\_TRIG |
| CLK | F\_TRIG |
| CU, R, PV | CTU\_INT, CTU\_DINT, CTU\_UINT, CTU\_UDINT |
| CU, LD, PV | CTD\_INT, CTD\_DINT, CTD\_UINT, CTD\_UDINT |

Продолжение таблицы 9.9

|  |  |
| --- | --- |
| CU, CD, R, LD, PV | CTUD\_INT, CTUD\_DINT, CTUD\_UINT, CTUD\_UDINT |
| IN, PT | TP |
| IN, PT | TON |
| IN, PT | TOF |

Вызов входного оператора завершает назначение параметров функционального блока. После этого передача значений в блок FB больше невозможна Теперь могут считываться лишь выходные значения.

Только команды загрузки и сохранения для текущего назначе­ния параметров блоков FB/DFB позволяются между первой ко­мандой загрузки фактического параметра и вызовом FB/DFB. Никакие другие команды не разрешаются в этом расположении.

Пример использования входных операторов приведен на рисунке 9.7.

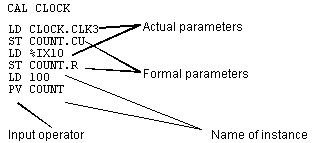


Рисунок 9.7 – Пример использования входных операторов

Команда CAL COUNT не разрешается, так как функциональный блок уже вызывается командой PV.

**Использование выхода функционального блока и блока DFB.**

Выходы FB/DFB могут всегда использоваться, если переменная (доступная только для чтения) также может использоваться.

Пример использования выхода функционального блока и блока DFB приведен на рисунке 9.8.

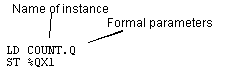


Рисунок 9.8 – Использование выхода функционального блока

Пример на рисунке 9.9 показывает, объявление и вызов функционального блока на языках IL и FBD.

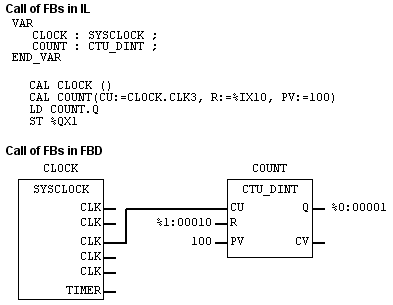


Рисунок 9.9 – Объявление и вызов функционального блока на языках IL и FBD

**Обращения к функциям.** Чтобы найти имена доступных функций, следует обратиться к библиотекам блоков.

Функции вызываются списком команд, состоящим из первого фактического параметра, загруженного в аккумулятор, и имени функции. В случае необходимости этот список сопровождается списком дополнительных фактических параметров. Важную роль играет порядок распечатки формальных параметров в обращении к функции. Список фактических параметров может иметь конец строки сразу же после запятой. После выполнения функции ре­зультат функции сохраняется в аккумуляторе и в операнде с по­мощью оператора ST или используется для прямой дальнейшей обработки.

Пример обращения к функции с фактическим параметром при­веден на рисунке 9.10, а пример обращения к функции с несколькими фактическими параметрами – на рисунке 9.11.

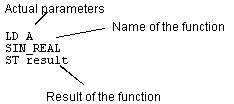


Рисунок 9.10 – Обращения к функции с фактическим параметром

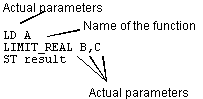


Рисунок 9.11 – Обращения к функции с несколькими фактическими параметрами

Функция может также вызываться списком команд, состоя­щим из первого фактического параметра, загруженного в аккумулятор, и имени функции. Этот список сопровождается спи­ском присвоения значений фактических параметров формаль­ным параметрам. Порядок распечатки формальных параметров в этом случае не имеет значения. Однако фактический параметр, загруженный с помощью оператора LD, продолжает быть первым параметром функции.

Пример обращения к функции с несколькими фактическими параметрами приведен на рисунке 9.12.

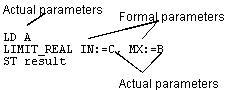


Рисунок 9.12 – Обращение к функции с несколькими фактическим параметрами

Если значение, которое будет обработано, находится уже в аккумуляторе, команда загрузки может быть опущена.

Пример обращения к функ­ции с обработкой значения ак­кумулятора:

LIMIT\_REAL B,C

ST result

Если обработка результата должна продолжиться немедленно, команда сохранения может быть опущена.

Пример обращения к функции с прямой непрерывной обработкой результата:

LD A

LIMIT\_REAL B,C

MUL E

Пример объявления и обращения к функции на языках IL и FBD приведен на рисунке 9.13

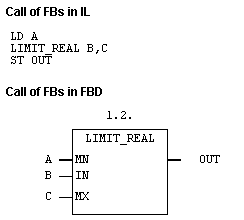


Рисунок 9.13 – Объявление и обращение к функциям на языках IL и FBD

**Комментарии.** В IL-редакторе комментарии начинаются с символьной цепочки «(\* и заканчиваются символьной цепочкой \*)». Любой комментарий может быть введен между этими двумя символьными цепочками. Комментарии отображаются в цвете.

Согласно стандарту комментарии позволяются только в конце строки. Стандарт не разрешает также вложение комментариев.

**Проверка синтаксиса и генерация объектного кода.** Генерация объектного кода будет выполнятся вместе с проверкой синтаксиса, когда секция закрывается. Любые найденные ошибки затем отображаются в окне сообщений.

Оптимизация кода во время выполнения программы достигается реализацией арифметики целого числа (например «+» или «-») через простые команды процессора вместо вызовов блоков EFB. Команды процессора выполняются намного быстрее, чем вызов блоков EFB, но они не генерируют сообщения об ошибках типа переполнения арифметики или массива. Эта опция должна использоваться, только если есть уверенность, что не имеется никаких арифметических ошибок в программе.

Пример самого быстрого кода:

LD in1

ADD 1

ST out1

**Опция Fastest code (restricted checking).** (Project → Code generation options). Если выбрать эту опцию, сложение in1+1 будет выполняться командой процессора add. В этом случае код выполняется быстрее, чем при вызове блока EFB ADD\_INT. Однако ошибка во время выполнения программы не будет сгенерирована, если in1 есть 32767. В этом случае разрядная сетка переменной out1 переполняется из 32767 в – 32768!

**Опция Unassigned parameters cause warnings.** (Option → Preferences → Analysis). Стандарт разрешает вызывать функции и функциональные блоки без распределения всех входных параметров. Этим неиспользуемым параметрам неявно присваивается 0 или они сохраняют значение из последнего вызова (только функциональные блоки).

Активизация флажка опции отобразит список неиспользуемых параметров в окне сообщения во время генерации объектного кода.

**Опция Enable Loop Control.** (Project → Code generation options) При активизации флажка осуществляется проверка внутри IL- и ST-секций для контуров, чтобы определить, можно ли эти контуры покинуть снова внутри некоторого периода времени. Разрешенное время для контуров зависит от определенного времени сторожа (Watchdog time). Разрешенное время всегда немного меньше, чем время сторожа. Использование этой опции предотвратит зацикливание программы на воспроизведение контура (бесконечный контур). Если бесконечный контур обнаруживается, обработка вызванной секции завершается и обрабатывается сразу следующая секция. Последующее сканирование обработает секцию снова, пока бесконечный контур не будет обнаружен или пока секция правильно не завершится.

**Интерактивные функции.** Редакторы языков IL и ST делают доступными три режима анимации:

* анимацию двоичных переменных;
* анимацию выбранных переменных;
* создание контролируемых полей.

Анимация прямых адресов и прямых входов/выходов блоков FB невозможна.

**Анимация двоичных переменных.** Анимация двоичных значений активизируется командой меню Online → Animate binary values. Этот режим отображает текущее сигнальное состояние двоичных переменных в окне редактора:

* переменная красного цвета имеет значение 0;
* переменная зеленого цвета имеет значение 1;
* переменная желтого цвета – не двоичная переменная.

**Анимация выбранных переменных.** Диалоговое окно, которое отображает текущие сигнальные состояния выбранных переменных, активизируется командой меню Online → Animate Selected. Для анимации выбранных переменных секция должна быть анимирована. Иначе следует активизировать анимацию с помощью команды Online → Animate binary values.

Выбираемые переменные и многоэлементные переменные идентифицируются в буквенном обозначении:

* переменная красного цвета имеет значение 0;
* переменная зеленого цвета имеет значение 1;
* переменная желтого цвета – не двоичная переменная.

Диалоговое окно отображает имя выбранной переменной или многоэлементной переменной, ее тип данных и текущее значение.

Диалоговое окно остается открытым, пока не будет закрыто или анимация не будет завершена. Если открывается несколько секций текстовых языков и в них вызывается диалоговое окно, то оно может быть открыто для каждой секции. Имя секции отображается в названии диалогового окна.

Несколько переменных или многоэлементных переменных могут быть вставлены в диалоговое окно. Для этого следует выбрать соответственно желаемые переменные или многоэлементные переменные и принять их с помощью команды Online → Animate Selected в диалоговое окно.

Пример 9.1. Согласно условиям примера 7.1 реализовать решение задачи на языке IL.

Решение на языке IL представлено ниже

VAR

FBI\_1\_1 : RS;

FBI\_1\_2 : RS;

FBI\_1\_3 : TON;

END\_VAR

LD vu

ST FBI\_1\_1.S

LD td

ST FBI\_1\_1.R1

CAL FBI\_1\_1

LD FBI\_1\_1.Q1

ST FBI\_1\_3.IN

LD t#3s

ST FBI\_1\_3.PT

CAL FBI\_1\_3

LD FBI\_1\_3.Q

ST td

LD FBI\_1\_3.Q

ST FBI\_1\_2.S

LD nu

STN FBI\_1\_2.R1

CAL FBI\_1\_2

LD FBI\_1\_2.Q1

ST zd