Дата проведения занятия 11 ноября 2020 г.

Номер пары: 55.

Группа: 21А

Тема занятия: Элементы FBD-схем.

Срок выполнения 14.11.2020

**По запросу преподавателя**, фото конспекта скинуть в «В контакте» Орлову А.А. (id421045327) личным сообщением.

Проверка освоения теоретического материала будет произведена выполнением проверочной работы после изучения темы.

**Задание.**

Используя предложенные справочные материалы (текст после вопросов) и другие источники информации (учебники, интернет), составить конспект по теме: Элементы FBD-схем.

**В конспекте обязательно должны быть выполнены задания и ответы на вопросы:**

1. Поясните, что представляет собой язык функциональных блоковых диаграмм FBD. Что необходимо учитывать при «работе» с программой, написанной на языке FBD.
2. Перечислите функции элементов программы, приведите 2-3 примера элементов, выполняющих операции, относящиеся к этим функциям. Кратко их поясните.
3. Перечислите функциональные блоки программы. Кратко их поясните.

**Общие сведения**

Главным требованием к программируемым логическим контроллерам (ПЛК) всегда была и остается возможность его эксплуатации существующим техническим персоналом. Здесь нужны простые и наглядные языки, позволяющие излагать задачу в близких к применяемым технологиям категориях. Сложный язык программирования снижает шансы *ПЛК* на широкое использование. Сначала каждая фирма, выпускающая *ПЛК,* разрабатывала для них свое программное обеспечение, и к 1979 году появилось множество языков программирования *ПЛК.* В результате прикладные программы, написанные для одного типа *ПЛК,* не могли быть использованы в *ПЛК* другой фирмы. Для решения проблем совместимости *ПЛК* различных фирм-изготовителей в рамках Международной Электротехнической Комиссии *(МЭК)* была создана специальная группа технических экспертов, которая занялась разработкой стандарта на языки программирования.

В стандарт включено пять языков программирования *ПЛК.* Включение в стандарт пяти языков объясняется в первую очередь историческими причинами. Вошедшие в стандарт языки созданы на основе наиболее популярных языков программирования, наиболее распространенных в мире контроллеров. Поэтому если взять любой контроллер, работающий в современном производстве, то его программу можно перенести в среду *МЭК61131* с минимальными затратами. Инженер, спроектировавший машину, должен иметь возможность самостоятельно написать программу управления. Никто лучше его не знает, как должна работать данная машина. Инженер, привыкший работать с электронными схемами, гораздо легче сможет выражать свои мысли в графических языках *LAD* или *FBD.* Если он знаком с языками *PASCAL* или С, то использование текстового языка *ST* не составит для него сложности.

***Язык функциональных блоковых диаграмм FBD (****Function Block Diagram) -* это графический язык программирования. Диаграмма *FBD* очень напоминает принципиальную схему электронного устройства жесткой логики на микросхемах (см. рис. 1). Программа на языке *FBD* не обязательно должна представлять большую единую схему, диаграмма образуется из множества цепей, которые выполняются одна за другой. *FBD-схемы* очень четко отражают взаимосвязь входов и выходов диаграммы. Если алгоритм изначально хорошо описывается с позиции сигналов, то его FBD-представление всегда получается нагляднее, чем в текстовых языках.

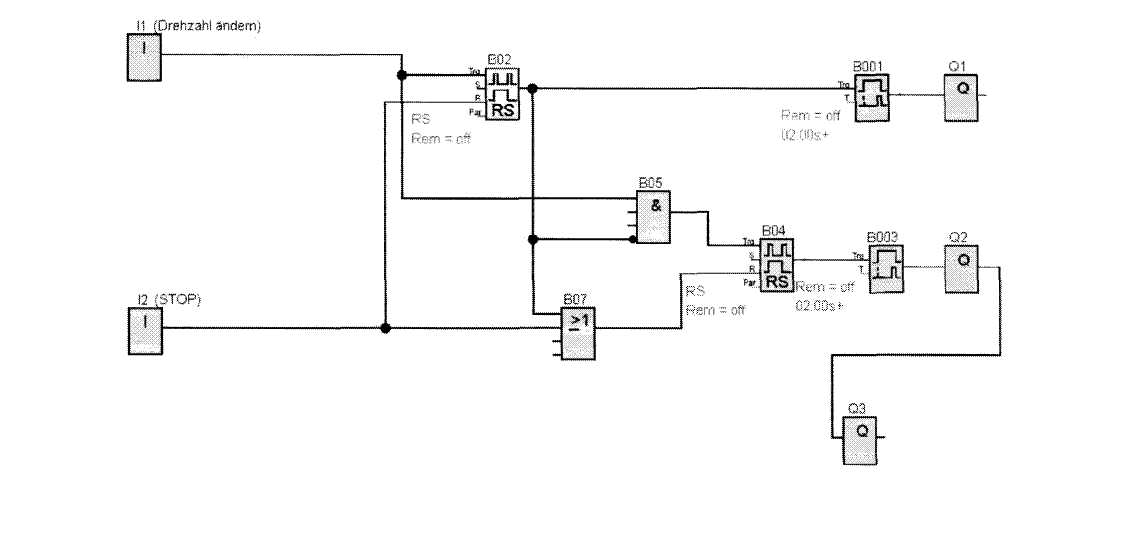


Рис. 1. Пример FBD-диаграммы

Привычная символика облегчает обращение с контроллером. «Проводники» в *FBD* могут проводить сигналы (передавать переменные) любого типа (логические, аналоговые, время и т.д.). Шины питания на *FBD* диаграмме не показываются. Выходы блоков могут быть поданы на входы других блоков либо непосредственно на выходы *ПЛК.* Сами блоки, представленные на схеме как «черные ящики», могут выполнять разные функции. В составе всех без исключения комплексов программирования присутствуют блоки, выполняющие логические операции («И», «ИЛИ», «НЕ»), функции триггеров, сдвиговых регистров, счетчиков импульсов, таймеров, пороговых выключателей, компараторов. Т.е. если научиться программировать ПЛК одного производителя, то в дальнейшем при работе с ПЛК других производителей, затруднения не должны возникать, т.к. в программах для них будут присутствовать блоки, выполняющие одинаковые действия. В тоже время следует помнить, что для получения информации о включенных в состав конкретного комплекса библиотечных компонентах необходимо использовать руководство по применению.

Рассмотрим элементы языка функциональных блоковых диаграмм FBD для программируемого реле ПР110. Для программирования которого применяется среда программирования OWEN Logic.

**Библиотека компонентов среды программирования OWEN Logic**

**Функции элементов программы**

1. **Битовые логические операции**

При использовании функций **И** и **ИЛИ** следует учитывать, что не подключенные входы логических элементов к входам программируемого реле (ПР) или другим элементам в программе, будут иметь следующие состояния:

* для элемента **И (AND) -** логическая «1»;
* для элемента **ИЛИ (OR) -** логический «0».

В этом случае логические элементы выполняют функцию повторителя сигнала. Для увеличения числа входов у логических элементов используется их каскадное включение, например, как это показано на рисунке 1.

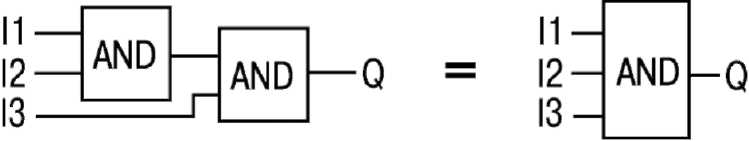
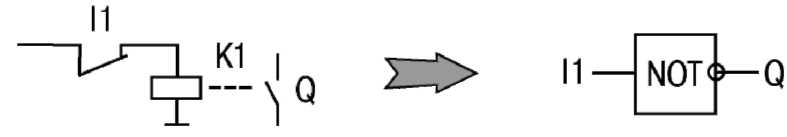


Рисунок 1. – Пример каскадного включения логических элементов И

**Функция «НЕ» (NOT)**

|  |  |
| --- | --- |
| ***I1*** | ***Q*** |
| 0 | 1 |
| 1 | 0 |

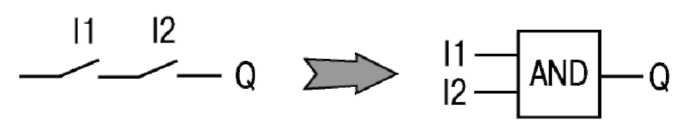


Элемент используется для инвертирования сигнала.

На выходе элемента логическая «1» (выход включен), если на входе логический «0» (контакты разомкнуты) и наоборот - сигнал инвертируется.

Если на входы функции заведены целочисленные значения, то операция будет произведена над каждым битом в отдельности.

**Функция «И» (AND)**



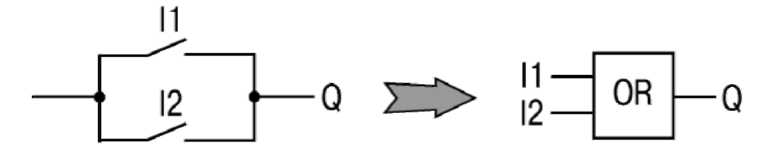
Элемент используется для выполнения логических операций. На выходе элемента логическая «1» (выход включен), если на всех входах логическая «1» (все входы включены - контакты замкнуты).

|  |  |  |
| --- | --- | --- |
| ***I1*** | ***I2*** | ***Q*** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Работе соответствует приведенная таблица состояний.

Если на входы функции заведены целочисленные значения, то операция будет произведена над каждым битом в отдельности.

**Функция «ИЛИ» (OR)**



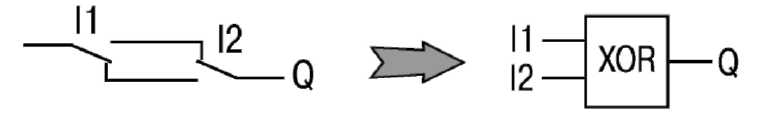
|  |  |  |
| --- | --- | --- |
| ***I1*** | ***I2*** | ***Q*** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Элемент используется для выполнения логических операций. На выходе элемента логическая «1» (выход включен), если хотя бы на одном входе логическая «1» (контакты замкнуты).

Работе соответствует приведенная таблица состояний.

Если на входы функции заведены целочисленные значения, то операция будет произведена над каждым битом в отдельности.

**Функция «исключающее ИЛИ» (XOR)**



|  |  |  |
| --- | --- | --- |
| ***I1*** | ***I2*** | ***Q*** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Элемент используется для выполнения логических операций. На выходе элемента логическая «1» (выход включен), только если на любом из входов логическая «1».

Работе соответствует приведенная таблица состояний.

Если на входы функции заведены целочисленные значения, то операция будет произведена над каждым битом в отдельности.

1. **Арифметические операции**

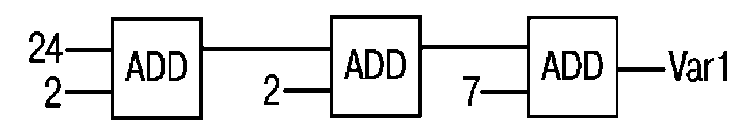
**Сложение (ADD)**



Результатом выполнения операции на выходе является сумма входных значений.

Если при выполнении операции значение числа получается больше 4294967295 (32 бита), выходящие за разрядность 32 бита биты отсекаются.

Пример применения элементов для сложения чисел 24 +2 + 4 + 7 = Var1:



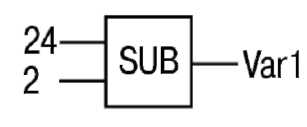
**Вычитание (SUB)**



Результатом выполнения операции на выходе является разность входных значений. Если при выполнении операции на первом входе (V1) значение числа меньше второго (V2), результатом будет число, полученное сложением младшего числа плюс 0x100000000 (4294967296) минус значение числа большего:

(V1 + 0x100000000) - V2 = Q.

Пример применения элемента для вычитания чисел 24 - 2 = Varl = 22:



Пример применения элемента для вычитания чисел, где V1 меньше V2:

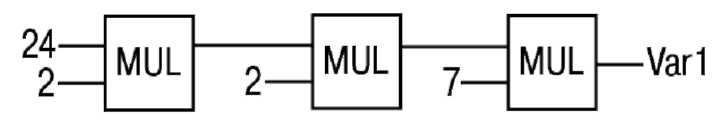
2 - 24 = Var1 = 4294967274.

**Умножение (MUL)**

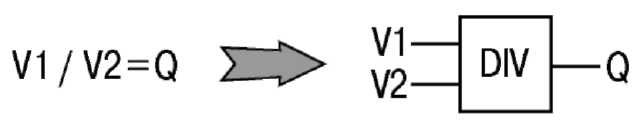


Результатом выполнения операции на выходе является перемножение входных значений. Если при выполнении операции значение числа получается больше 4294967295 (32 бита), выходящие за разрядность 32 бита биты отсекаются.

Пример применения элементов для перемножения чисел 24∙2∙2∙7 = Varl:

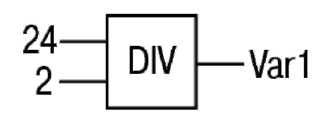


**Деление (DIV)**

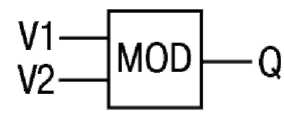


Результатом выполнения операции на выходе является деление входных значений. Если в результате деления получаются доли целого числа, то на выходе производится округление значения до целого числа в меньшую сторону. При делении на 0 на выходе элемента будет 0xFFFFFFFF.

Пример применения элемента для деления чисел 24/2 = Varl:

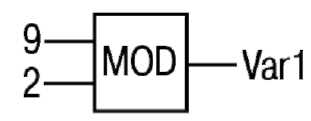


**Деление с остатком (MOD)**



Результатом выполнения операции на выходе является остаток от деления входных значений.

Пример применения элемента для выделения целого остатка от деления чисел 22 на 3 (Var1 = 1):



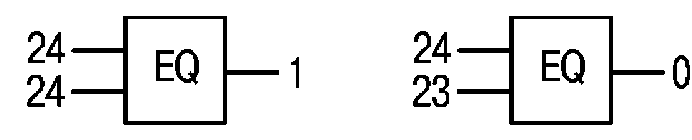
1. **Операции сравнения и выбора**

**Равно (EQ)**

|  |  |
| --- | --- |
|  | **V1 = V2; Q = 1**  **V1 > V2; Q = 0**  **V1 < V2; Q = 0** |

Результатом выполнения операции на выходе является лог. «1», если входные числа равны.

Примеры применения элементов:

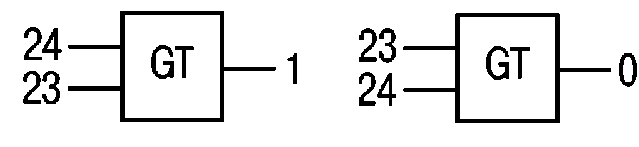


**Больше (GT)**

|  |  |
| --- | --- |
|  | **V1 = V2; Q = 0**  **V1 > V2; Q = 1**  **V1 < V2; Q = 0** |

Результатом выполнения операции на выходе является лог. «1», если входное число на верхнем входе (V1) больше числа на втором входе (V2).

Примеры применения элементов:

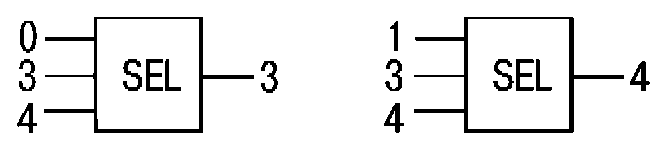


**Выбор (SEL)**

|  |  |
| --- | --- |
|  | **V1 = 0; Q = V2**  **V1 = 1; Q = V3** |

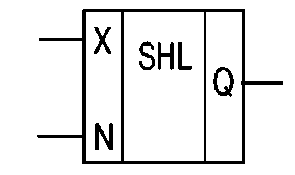
Результатом выполнения операции на выходе является входной сигнал V2, если V1 равен логическому «0», и V3, если V1 равен лог «1».

Примеры применения элементов:



**Операции логического битового сдвига и преобразования**

**Побитный логический сдвиг влево (SHL)**

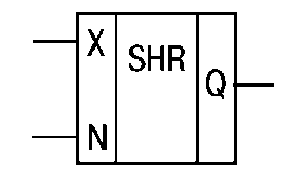


Элемент используется для выполнения операции побитного логического сдвига операнда X влево на N бит с дополнением нулями справа.

Пример применения для сдвига числа 38(десятичное) = 00100110(двоичное):

|  |  |
| --- | --- |
|  |  |

**Побитный логический сдвиг вправо (SHR)**

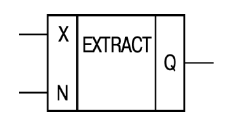


Элемент используется для выполнения операции побитного логического сдвига операнда X вправо на N бит с дополнением нулями слева.

Пример применения для сдвига числа 152(десятичное) = 10011000(двоичное):

|  |  |
| --- | --- |
|  |  |

**Чтение бита (EXTRACT)**

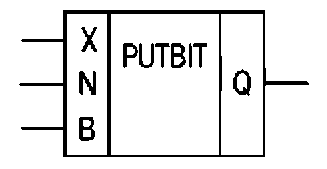


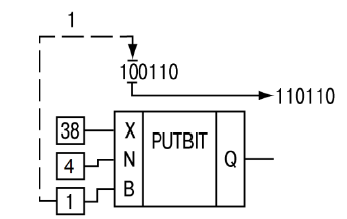
Элемент используется для выполнения операции чтения значения бита N в числе на входе X. Биты нумеруются с нуля. Число на входе Х задается в десятичной системе. Выходное значение всегда двоичное (0 или 1).

Пример применения для чтения пятого бита из числа 81(десятичное) = 1010001(двоичное):

|  |  |
| --- | --- |
|  |  |

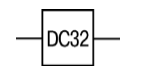
**Запись бита (PUTBIT)**



Элемент используется для выполнения операции записи в числе X значения бита N в состояние, указанное на входе B (лог. «0» или «1»). Число на входе Х задается в десятичной системе. Выходное значение всегда двоичное (0 ли 1).

Пример применения записи четвертого бита в лог. «1» для числа 38(десятичное) = 100110(двоичное):

**Дешифратор (DC32)**



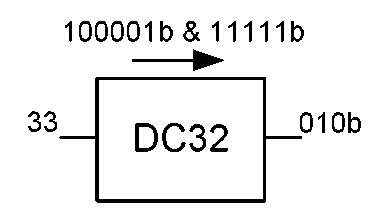
Элемент используется для выполнения операции преобразования двоичного кода на входе в позиционный код на выходе.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Двоичный код** | | | | | **Позиционный код** | | | | | | | | |
| **5** | **4** | **3** | **2** | **1** | **32** | **31** | … | **6** | **5** | **4** | **3** | **2** | **1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | … | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | … | 0 | 0 | 1 | 0 | 0 | 0 |
|  |  | 1 | 0 | 0 | 0 | 0 | … |  | 1 |  |  |  |  |
| … | | | | | … | | | | | | | | |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | … | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | … | 0 | 0 | 0 | 0 | 0 | 0 |

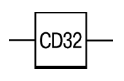
Перед выполнением операции над значением на входе предварительно выполняется побитовая логическая операция «И» с операндом 0x1F (11111b).

Работе элемента соответствует приведенная таблица состояний.

Пример применения:



**Шифратор (CD32)**



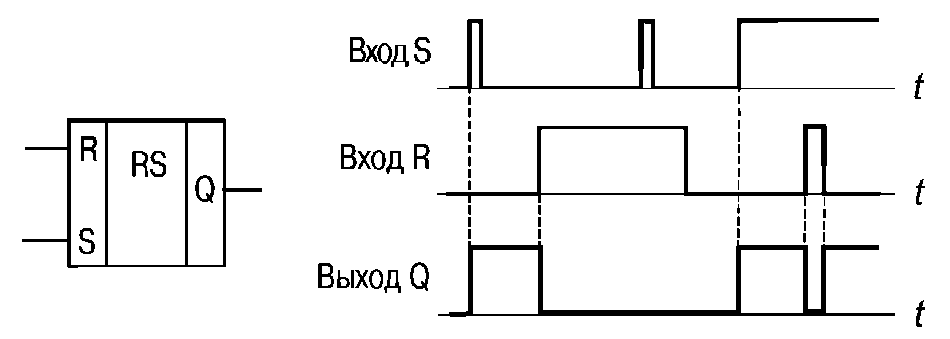
Элемент используется для выполнения операции преобразования позиционного кода на входе в двоичный код на выходе.

В случае, если входное двоичное значение имеет более одной логической «1» в разрядах, работа элемента ведется только со старшим единичным разрядом.

Работе элемента соответствует приведенная таблица состояний в пункте «Дешифратор».

**Функциональные блоки программы**

**RS-триггер с приоритетом выключения (RS)**

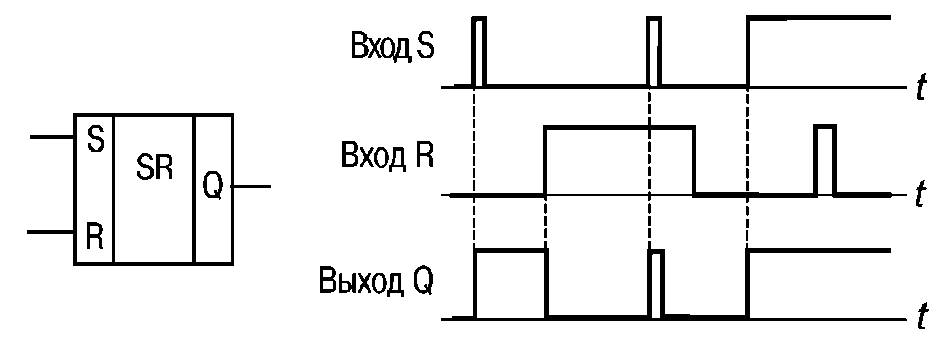


Блок используется для переключения с фиксацией состояния при поступлении коротких импульсов на соответствующий вход. Работу поясняет приведенная на рисунке диаграмма.

На выходе блока появится логическая «1» по фронту сигнала на входе S.

При одновременном поступлении сигналов на входы приоритетным является сигнал входа R.

**SR-триггер с приоритетом включения (SR)**

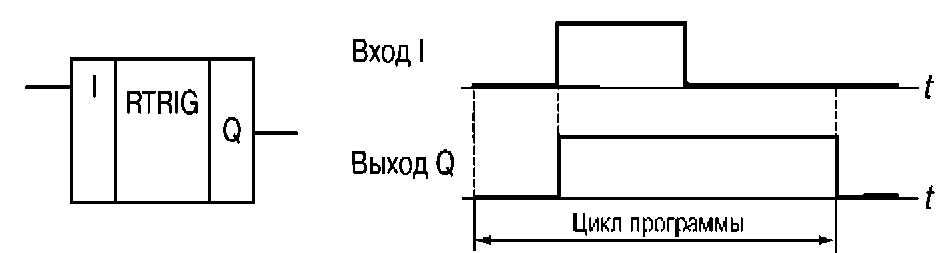


Блок используется для переключения с фиксацией состояния при поступлении коротких импульсов на соответствующий вход. Работу поясняет приведенная на рисунке диаграмма.

На выходе блока появится логическая «1» по фронту сигнала на входе S.

При одновременном поступлении сигналов на входы приоритетным является сигнал входа S.

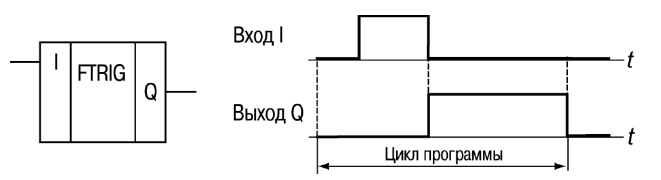
**Детектор переднего фронта импульса (RTRIG)**



Блок используется в случае необходимости иметь реакцию на изменение состояния дискретного входного сигнала. На выходе генерируется единичный импульс по переднему фронту входного сигнала.

Работу поясняет приведенная на рисунке диаграмма.

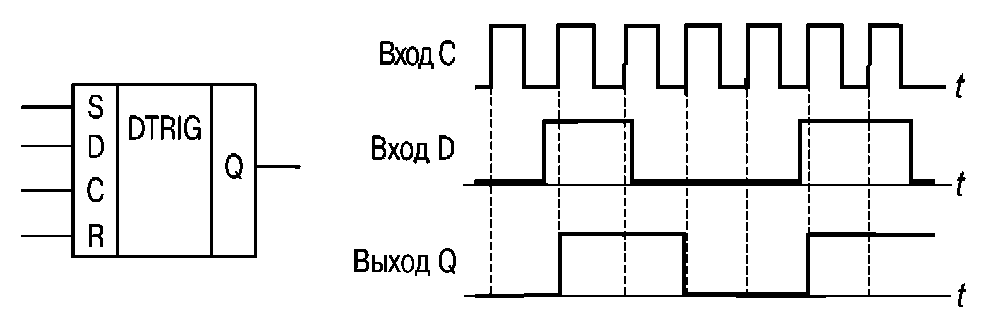
**Детектор заднего фронта импульса (FTRIG)**



Блок используется в случае необходимости иметь реакцию на изменение состояния дискретного входного сигнала. На выходе генерируется единичный импульс по заднему фронту входного сигнала.

Работу поясняет приведенная на рисунке диаграмма.

**D-триггер (DTRIG)**



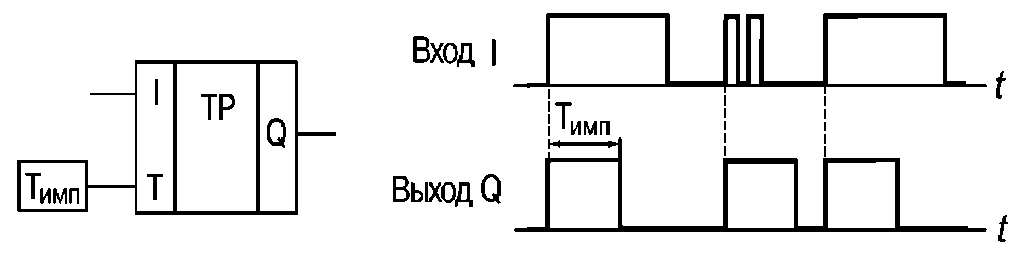
Блок используется для формирования импульса включения выхода на интервал времени импульса на входе D, при этом выходной интервал будет синхронизирован с тактовой частотой на входе С.

На выходе Q блока появится логическая «1» по фронту тактовых импульсов на входе С при наличии логической «1» на входе D. Возврат выхода в «0» произойдет по фронту тактовых импульсов на входе С при логическом «0» на входе D.

Работу поясняет приведенная на рисунке диаграмма.

Вход S обеспечивает принудительную установку выхода Q в состояние логической единицы; вход R является приоритетным и обеспечивает установку выхода Q в состояние логического «0».

**Импульс включения заданной длительности (ТР)**

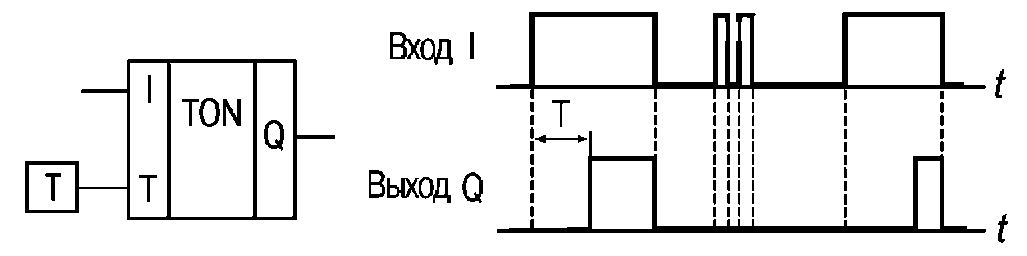


Блок используется для формирования импульса включения выхода на заданный интервал времени. Работу поясняет приведенная на рисунке диаграмма.

На выходе Q блока появится логическая «1» по фронту входного сигнала (I). После запуска выход Q не реагирует на изменение значения входного сигнала на интервале Тимп, а по истечении этого интервала сбрасывается в «0».

Допустимый диапазон значений Тимп = Т - от 0 до 4147200000 мс, или **48** дней.

**Таймер с задержкой включения (TON)**

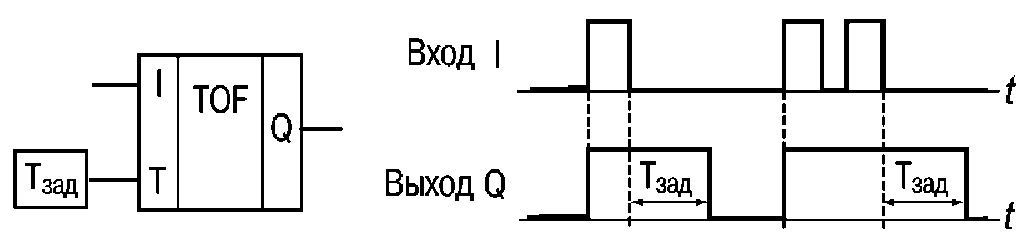


Блок используется для операции задержки передачи сигнала. Работу поясняет приведенная на рисунке диаграмма.

На выходе Q блока появится логическая «1» с задержкой относительно фронта входного сигнала (I). Выход включается логической «1» на входе продолжительностью не менее длительности Т, а выключается по спаду входного сигнала.

Допустимый диапазон значений Т - от 0 до 4147200000 мс, или **48** дней.

**Таймер с задержкой отключения (TOF)**

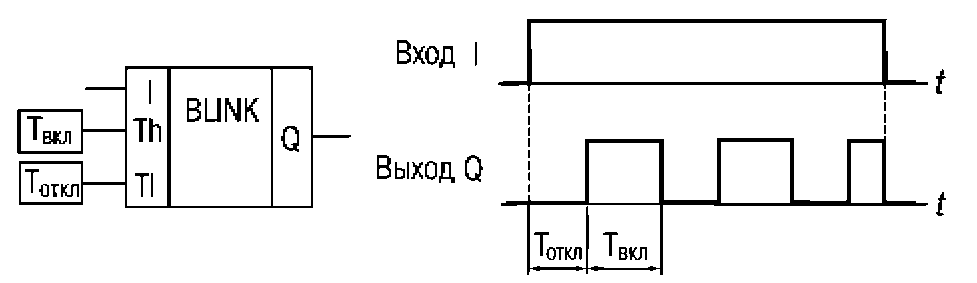


Блок используется для задержки отключения выхода. Работу поясняет приведенная на рисунке диаграмма.

На выходе блока появится логическая «1» по фронту сигнала на входе I, а начало отсчета времени задержки отключения (Тзад) происходит по каждому спаду входного сигнала. После отключения входного сигнала на выходе появится логический «0» с задержкой Тзад.

Допустимый диапазон значений Тзад - от 0 до 4147200000 мс, или **48** дней.

**Генератор прямоугольных импульсов (BLINK)**

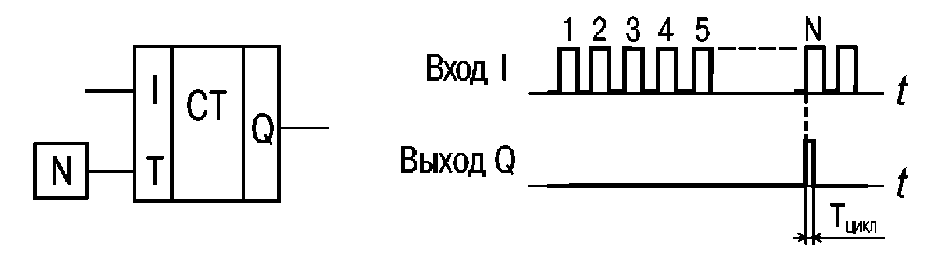


Блок используется для формирования прямоугольных импульсов пульсации.

На выходе Q генератора формируются импульсы с заданными параметрами длительности включенного (Твкл - логическая «1») и отключенного (Тоткл -логический «0») состояния на время действия управляющего сигнала на входе I (логической «1»). Работу поясняет приведенная на рисунке диаграмма.

Допустимый диапазон значений Твкл и Тоткл - от 0 до 4233600000 мс, или 49 дней.

**Инкрементный счетчик с автосбросом (CT)**

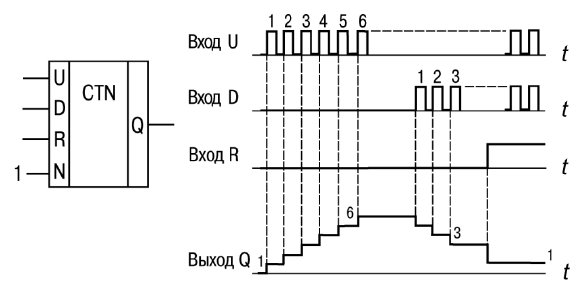


Блок используется для подсчета заданного числа импульсов N (вход Т -уставка числа импульсов). Работу поясняет приведенная на рисунке диаграмма.

На выходе Q счетчика появится импульс логической «1» с длительностью рабочего цикла прибора Тцикл), если число приходящих на вход I импульсов достигнет установленного значения N (на входе Т).

Допустимый диапазон значений числа импульсов N - от 0 до 65535.

**Универсальный счетчик (CTN)**



Блок является счетчиком, который используется для прямого и обратного счета. Операция «прямой счет» выполняется по переднему фронту импульса на входе прямого счета (U), что увеличивает значение выходного сигнала «Q». Импульсы, приходящие на вход D («обратный счет»), уменьшают значение выхода «Q».

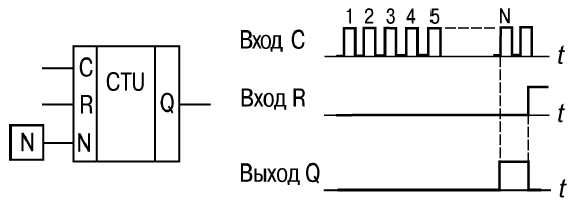
При поступлении на вход R логической «1», выход счетчика «Q» устанавливается в значение входа N.

Работу блока поясняет приведенная на рисунке диаграмма.

При одновременном поступлении сигналов на входы U и D приоритетным является сигнал входа U.

Допустимый диапазон значений числа импульсов N - от 0 до 65535.

**Инкрементный счетчик (CTU)**

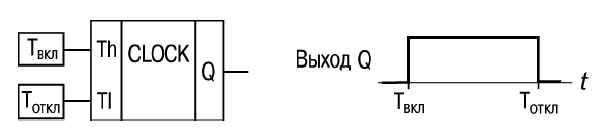


Блок используется для подсчета числа импульсов, приходящих на вход С. Работу поясняет приведенная на рисунке диаграмма - на выходе Q счетчика появится импульс лог. «1», если число приходящих на вход импульсов достигнет установленного значения на входе N (N - уставка).

Допустимый диапазон значений числа импульсов N - от 0 до 65535.

Счетчик сбрасывается в 0 по переднему фронту импульса на входе R. При одновременном поступлении сигналов на входы приоритетным является сигнал входа R.

**Интервальный таймер (CLOCK)**



Блок используется для формирования импульса включения выхода по часам реального времени. Работу поясняет приведенная на рисунке диаграмма.

Время включения (Твкл) и отключения (Тоткл) выходов устанавливают в качестве параметров блока.

Допустимый диапазон значений Твкл и Тоткл - от 0,00 с до 24 ч.

В случаи если значение время отключения задано раньше времени включения (рисунок 2), диаграмма переключений будет иметь вид, приведенный на рисунке 3 (выход будет включен до момента времени отключения).

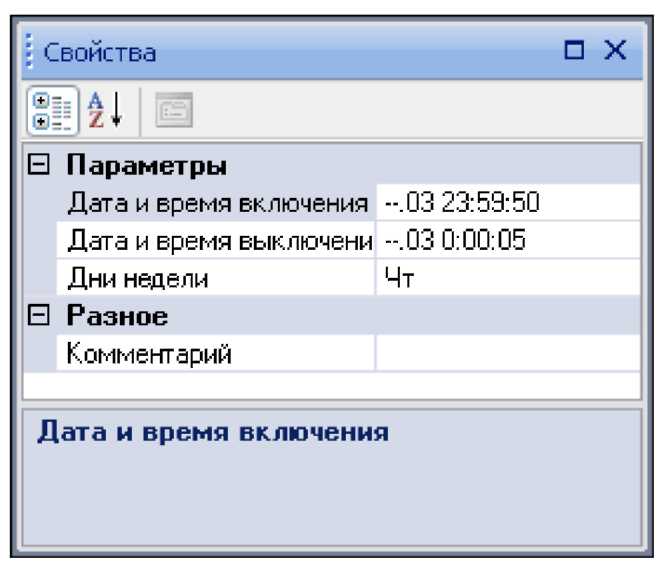


Рисунок 2 - Вкладка «Свойства» для настройки временных параметров работы блока

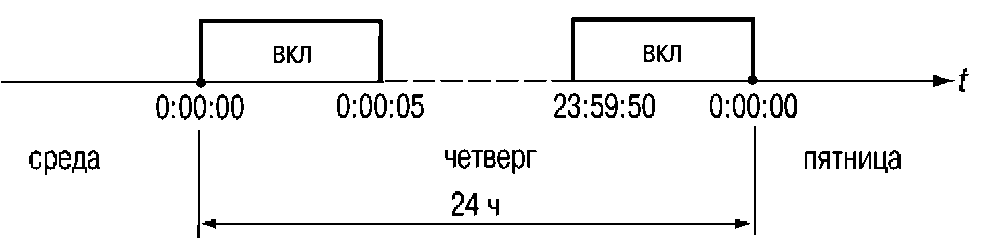
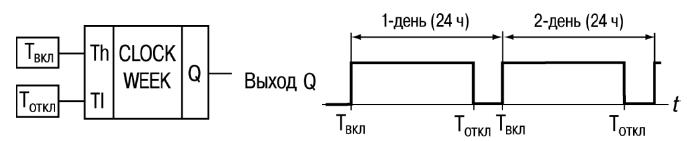
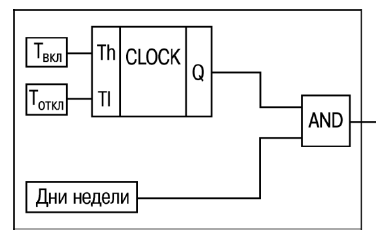


Рисунок 3 - Диаграмма включения выхода блока

**Интервальный таймер с недельным циклом (CLOCK WEEK)**



Блок используется для формирования импульса включения выхода по часам реального времени, с учетом дней недели. Работу поясняет приведенная на рисунке диаграмма. Внутренняя структура блока имеет вид:



Время включения (Твкл), отключения (Тоткл) выхода и дни недели работы устанавливают в качестве параметров блока.

Допустимый диапазон значений Твкл и Тоткл - от 0,00 с до 24 ч.