SIMATIC

S7-200 Примеры

Группа	Тема
3	**Modbus RTU Slave для S7-214**

Краткое описание

Данный пример включает группу подпрограмм и программ обработки прерываний для использования S7-214 в режиме свободнопрограммируемого интерфейса в качестве Modbus RTU slave. Данная программа поддерживает следующие функции Modbus:

1 Чтение выходов
2 Чтение входов
3 Чтение управляющих регисторов (V память)
4 Чтение входных регисторов
5 Запись одного выхода
6 Запись одного управляющего регистра
15 Запись нескольких выходов
16 Запись нескольких управляющих регистров

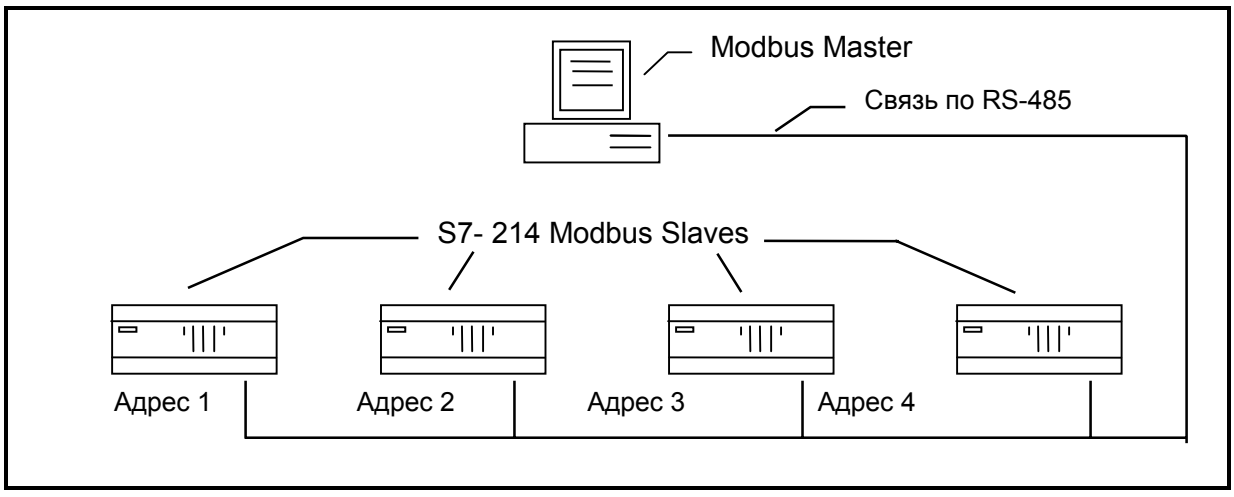


Рисунок 41.1

Структура программы

Драйвер протокола Modbus состоит из группы подпрограмм и программ обработки прерываний, которые инициализируют и обрабатывают запросы Modbus. Программа пользователя состоит из двух сегментов. Один сегмент инициализирует в первом цикле драйвер Modbus. Другой сегмент контролирует M-бит и обрабатывает возникающие запросы Modbus. Этот второй сегмент д.б. помещен в конце программы пользователя (непосредственно перед MEND), так чтобы данные изменялись только в конце цикла.

Список используемых подпрограмм и прерываний:

SBR 50	Инициализация драйвера Modbus RTU
SBR 51	Обработка запроса Modbus и передача ответа
SBR 52	Обработка функции Modbus 1 и 2
SBR 53	Обработка функции Modbus 3 и 4
SBR 54	Обработка функции Modbus 5
SBR 55	Обработка функции Modbus 6
SBR 56	Обработка функции Modbus 15
SBR 57	Обработка функции Modbus 16
SBR 61	Генерация ошибки отклика 2
SBR 62	Инициализация таблицы CRC
SBR 63	Подсчет CRC
INT 120	Обработка тайм-аута свободной линии
INT 121	Обработка символов пока ожидание тайм-аута свободной линии
INT 122	Прием первого символа (поле адреса) запроса
INT 123	Прием остальной части запроса
INT 124	Прерывание запроса после тайм-аута свободной линии
INT 125	Сброс поиска свободной линии после завершения передачи

Описание программы

Данная программа позволяет подключить один или больше S7-214 к главной станции Modbus. Программа использует функции свободнопрограммируемого интерфейса S7-214 для реализации протокола Modbus RTU. Modbus RTU - протокол master-slave; это означает, что структура сети состоит из одной активной станции (главной ЭВМ) и одной или нескольких пасивных станций. Каждая пасивная станция имеет уникальный адрес. Активная станция посылает запрос оной из пасивных станций и ожидает ответа от нее. Пасивная станция отвечает на полученный запрос или возникает ошибка. Если запрос был принят неверно, то возникает ошибка передачи, такая как ошибка четности или неверная CRC (контрольная сумма); пассивная станция не отвечает и активная станция должна повторить запрос после соответствующего времени ожидания.

Modbus RTU - двоичный протокол. Начало телеграммы обозначается временем свободной линии для 3.5 байт на текущей скорости обмена. Конец телеграмы так же обозначается тем же временем свободной линии. Т.к. время свободной линии - складывается из времени передачи символов, оно сильно зависит от скорости передачи. Программа, описанная ниже устанавливает время свободной линии равное значению соответствующему при 9600 Бод. Если скорость обмена меняется, то время свободной линии также д.б. изменено. Это описано в SBR 50.

Протокол Modbus RTU передает данные в 8 битном двоичном коде. Каждый символ также включает один стартовый бит, один или два стоповых бита (S7-214 поддерживает один стоповый бит) и опционально бит четности. Программа, описанная ниже, использует S7-214 в режиме 9600 Бод с четностью. Эти значения м.б. изменены в установках порта в SBR 50.

Протокол Modbus RTU использует CRC (Cyclical Redundancy Check - контроль при помощи циклического избыточного кода) для обнаружения ошибок. Данный пример использует таблицу со значениями CRC для ускорения вычисления CRC при проверки полученной телеграммы и

передачи ответа. Таблица CRC создается в верхней V памяти во время инициализации драйвера Modbus и требует около 700 милисекунд. Это происходит только во время первого цикла.

Для поддержки функций Modbus 1, 2, 3, 4, 5, 6, 15 и 16 используются соответствующие подпрограммы. Если конкретная активная станция не использует всех этих функций, их можно убрать, чтобы увеличить память доступную программе пользователя. Для этого необходимо убрать соответствующую подпрограмму и ее вызов. Все вызовы находяться в SBR 51.

Поддерживаются следующие функции:

1 Чтение состояния одного/нескольких выходов
- Возвращает состояние включено/выключено любого числа выходов (Q). Максимальное число выходов м.б. задано пользователем (см. ниже).

2 Чтение состояния одного/нескольких входов
- Возвращает состояние включено/выключено любого числа входов (I). Максимальное число входов м.б. задано пользователем (см. ниже).

3 Чтение одного/нескольких управляющих регистров
- Возвращает содержимое V памяти. Подразумевается, что управляющий регистр д.б. переменной для Modbus типа слово. Эта область начинается с V0. Размер (в словах) входов м.б. задан пользователем (см. ниже).

4 Чтение одного/нескольких входных регистров
- Чтение аналоговых входов. Эта функция возвращает содержимое области V памяти, отделенное от управляющих регистров. Пользователь должен добавить команды чтения слов AI и скопировать их, если необходимо, в V память. Область V памяти, используемая для этих команд, м.б. задана пользователем (см. ниже).

5 Управление одним выходом
- Запись регистра отображения выхода (Q). Выход, в действительности, не управляется, а только записывается.

6 Запись одного управляющего регистра
- Запись слова в V память.

15 Управление несколькими coil (выходами)
- Запись нескольких выходов (Q). Область выходов должна начинаться на границе байта (например, Q0.0 или Q2.0) и число записываемых выходов д.б. кратно восьми. Это не требования Modbus, а сделано для простоты реализации. Выхода, в действительности, не управляются, а только записывается регистр отображения выходов.

16 Запись нескольких управляющих регистров
- Запись нескольких слов в V память. За один запрос м.б. записано до 60 слов.

Следующие области памяти используются для конфигурирования драйвера Modbus. Эти области инициализируются в SBR 50 и м.б. изменены пользователем, чтобы изменить размер области памяти, доступной главной ЭВМ через драйвер Modbus.

VW3290 Максимальное число входов/выходов, доступных через Modbus.
Имеет значение для функций 1, 2, 5 и 15. Значение по умолчанию - 64 (I0.0 - I7.7 и Q0.0 - Q7.7).

VW3294 Максимальное число входных регистров, доступных через Modbus функцию 4. Значение по умолчанию - 16.

VW3296 Адрес в V памяти, куда будут при ответе считаны AI для Modbus функции 4. Значение по умолчанию - VB2000.

VB4095 Modbus адрес пассивной станции. Значение по умолчанию - адрес 1.

Следующие области V памяти используются драйвером Modbus и *не* должны изменяться пользователем.

M31.7 Меркер, используемый для сигнализации, что был получен запрос Modbus.

VB3300 - VB3559 Буфер драйвера

VB3560 - VB3575 Различные области рабочей памяти для Modbus

VB3580 - VB4091 Таблица данных CRC

Подпрограммы 50 - 63 зарезервированы для драйвера Modbus.

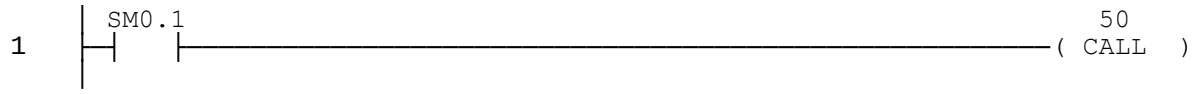
Прерывания 120 - 127 зарезервированы для драйвера Modbus.

Метки 254 и 255 зарезервированы для драйвера Modbus.

LAD (S7-MicroDOS)	STL (IEC)

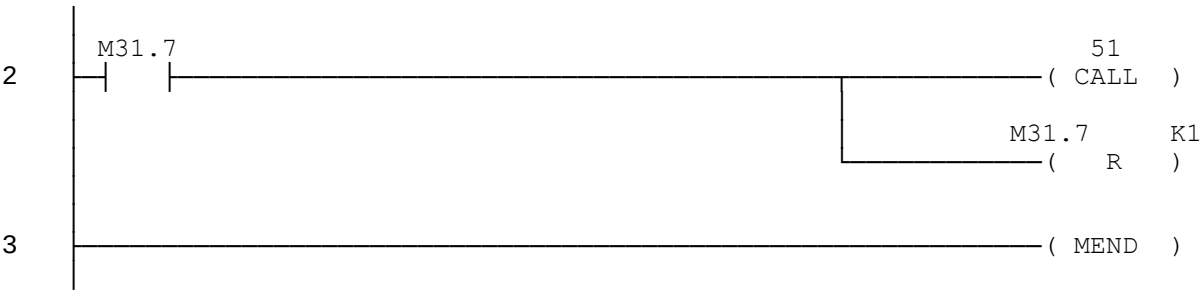
Главная программа

```
// Вызов подпрограмм инициализации драйвера Modbus во время первого цикла.
// Этот сегмент должен находиться в начале программы.
```




```
LD      SM0.1       // Если первый цикл,
CALL    50          // инициализация драйвера Modbus
```

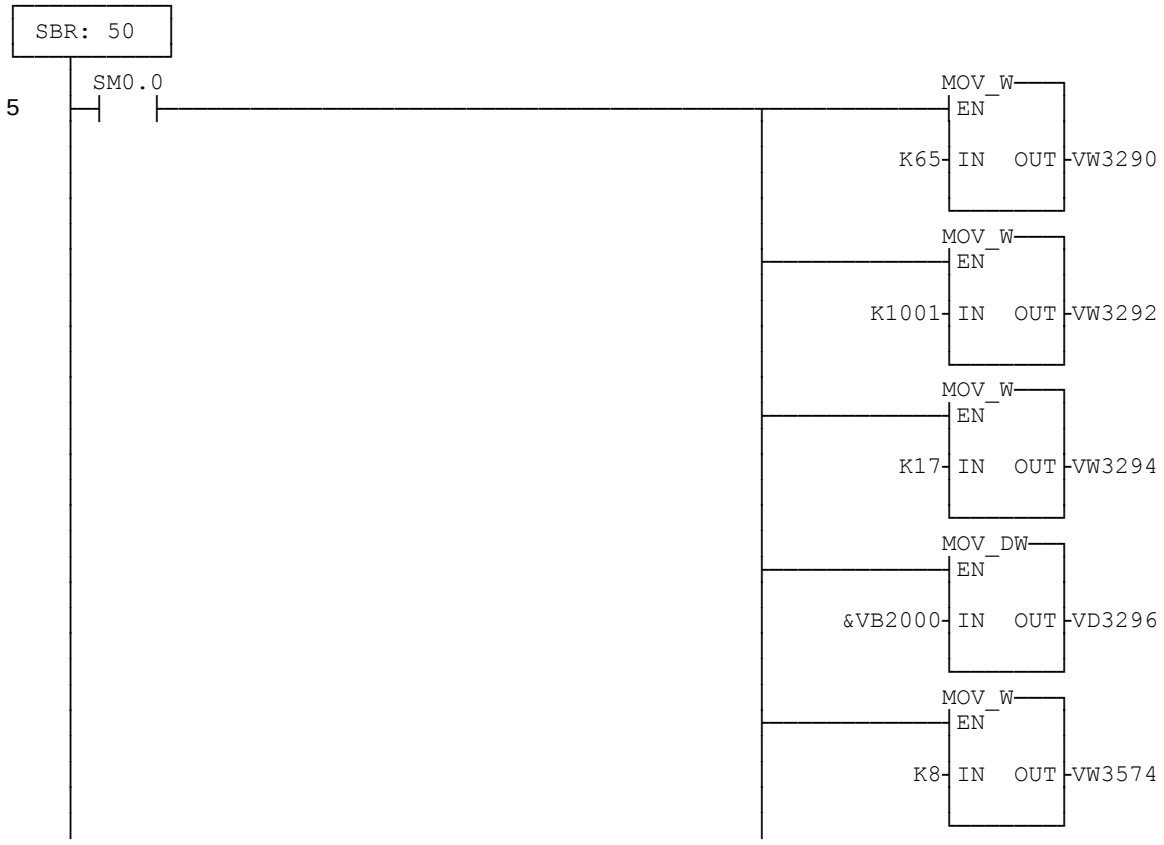
```
// Проверка получен ли запрос Modbus.  Если получен, то программа
// обработки полученного прерывания устанавливает M31.7.
// Этот сегмент должен находиться в конце программы.
```




```
LD      M31.7       // Если есть Modbus запрос,
CALL    51          // вызов обработчика и
R       M31.7,1     // сброс меркера запроса

MEND
```

Подпрограммы

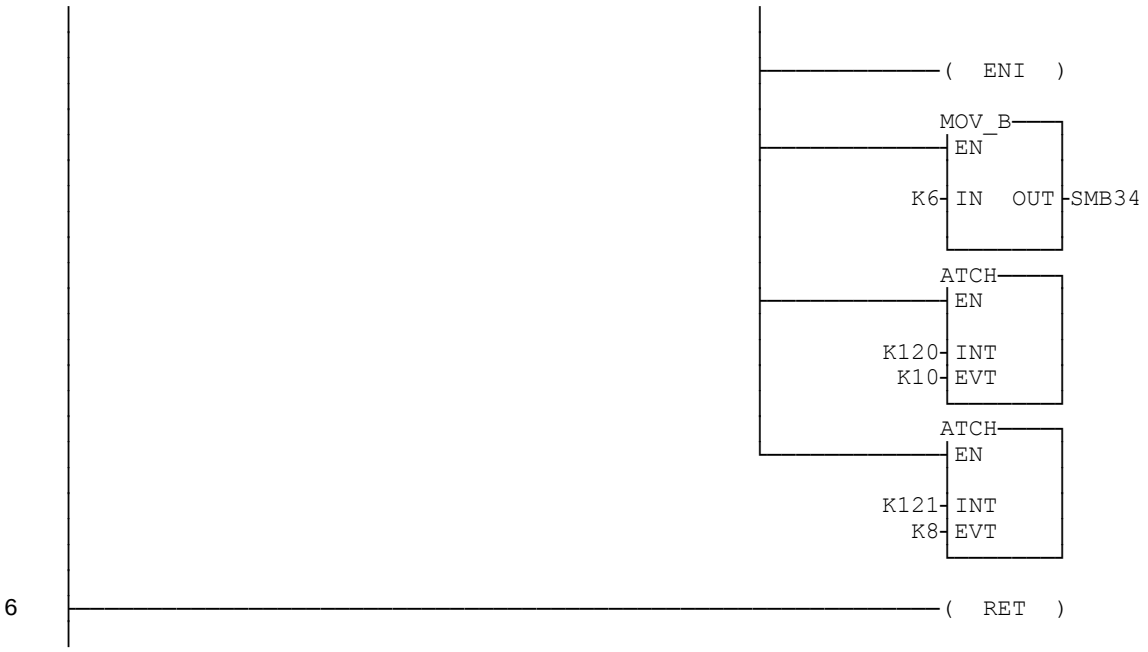
```
// Подпрограмма 50
//
// Инициализация драйвера Modbus для порта 0.
// ЗАМЕЧАНИЕ: Программа инициализации требует для исполнения около 690 милисек.
//  для инициализации таблицы CRC. Не беспокойтесь !
```




```
// Инициализация границ памяти для различных типов данных.
//
// ЗАМЕЧАНИЕ: Максимальное значение в памяти д.б. на единицу больше, чем граничное
//                    значение. Пример: Чтобы разрешить 32 выхода, используйте значение 33.

SBR         50
LD          SM0.0
MOVW        65,VW3290              // макс_регистр_отображения = 64 бита
MOVW        1001,VW3292            // макс. V слов = 2000 байт (1000 слов)
MOVW        17,VW3294              // макс. AI слов = 16 слов
MOVD        &VB2000,VD3296         // начало AI слов в V памяти
MOVW    8, VW3574                  // загрузить константу = 8 для использования
                                   // в математических операциях
```


```
// Инициализация адреса Modbus.

MOVB        1,VB4095               // Адрес Modbus = 1
```

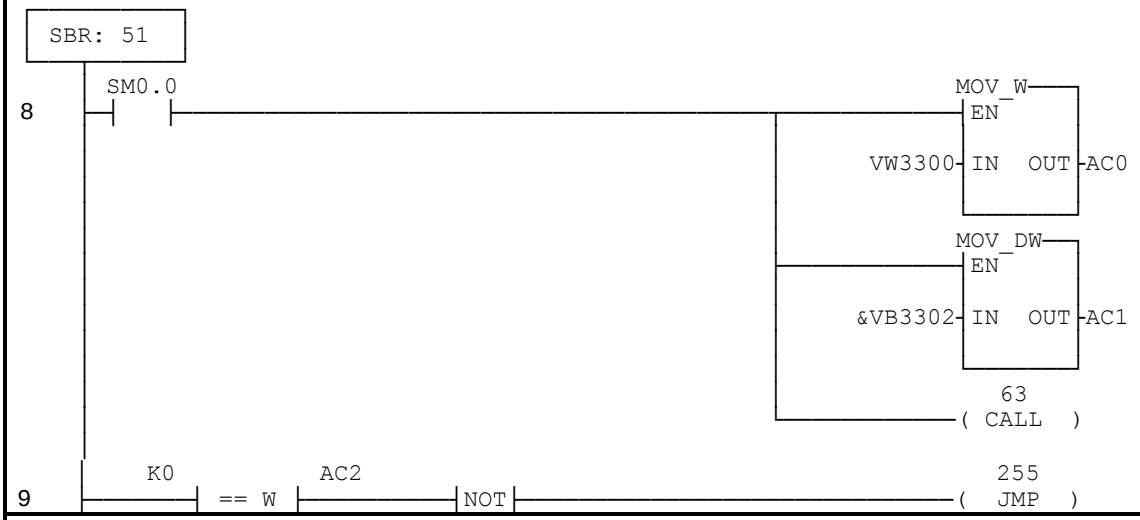

```
// Инициализация порта. См. Руководство по программированию Step-7 Micro
// для других параметров порта (например, скорости передачи, четности). RTU Modbus
// использует 8 битов данных. Скорость передачи и четность м.б. изменены.
//
// ЗАМЕЧАНИЕ: При установке 38.4K Бод функции работают неправильно.

MOVB    16#49,SMB30          // 9600 бод, 8 бит, с четностью

// Инициализация таблицы данных CRC и сброс меркера "продолжающаяся телеграмма".

CALL    62                   // Инициализация таблицы CRC Modbus
R       M31.7,1              // сброс меркера "продолжающаяся телеграмма"
```




```
// Телеграммы Modbus отделяются временем свободной линии для как минимум 3.5 байт,
// которое для 9600 Бод = 4 милисекунд. Время свободной линии устанавливается
// равным 6 милисек., чтобы гарантировать по крайней мере 5 милисек. (4 милисек. -
// время свободной линии + 1 милисек. для приема символа.
//
// ЗАМЕЧАНИЕ: Время тайм-аута д.б. изменено для разных скоростей передачи:
//
//          300 бод              166  милисек.
//          600                  84
//          1200                 43
//          2400                 22
//          4800                 12
//          9600                 6
//          19.2K                5

ENI                              // разрешение прерываний
MOVB    6,SMB34                  // установить таймер свободной линии > 5 милисек.
ATCH    120,10                   // начальный поиск свободной линии
```

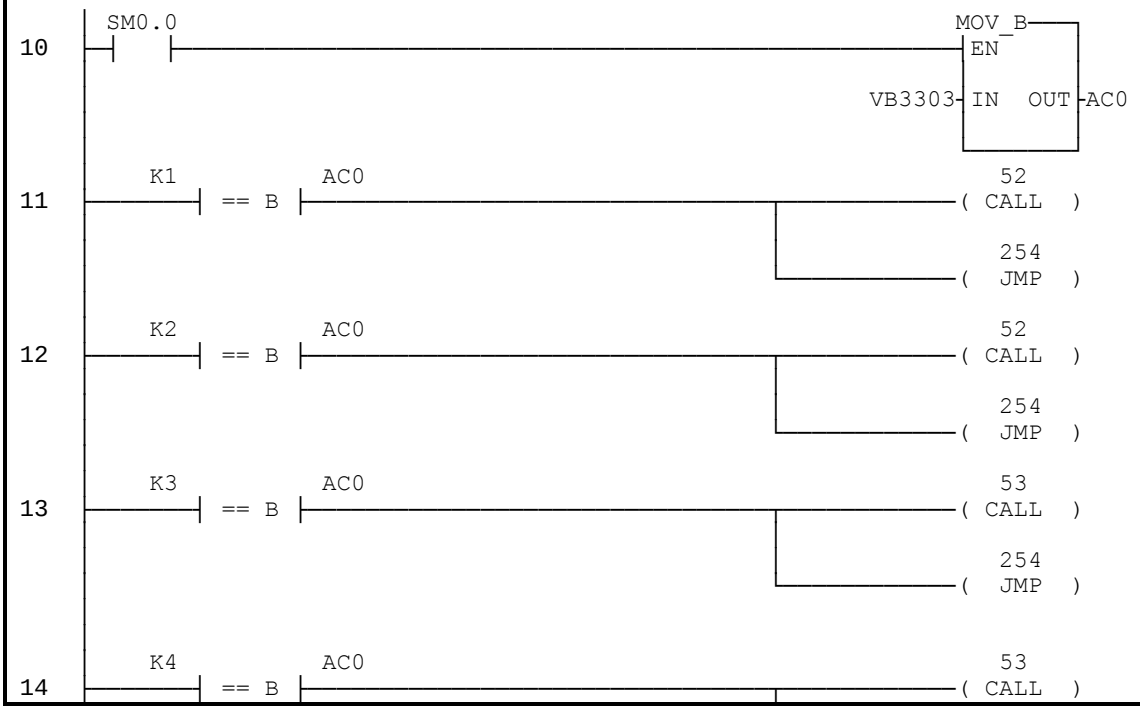
```
ATCH    121,8          // INT 121, если принят символ

RET                    // возврат
```

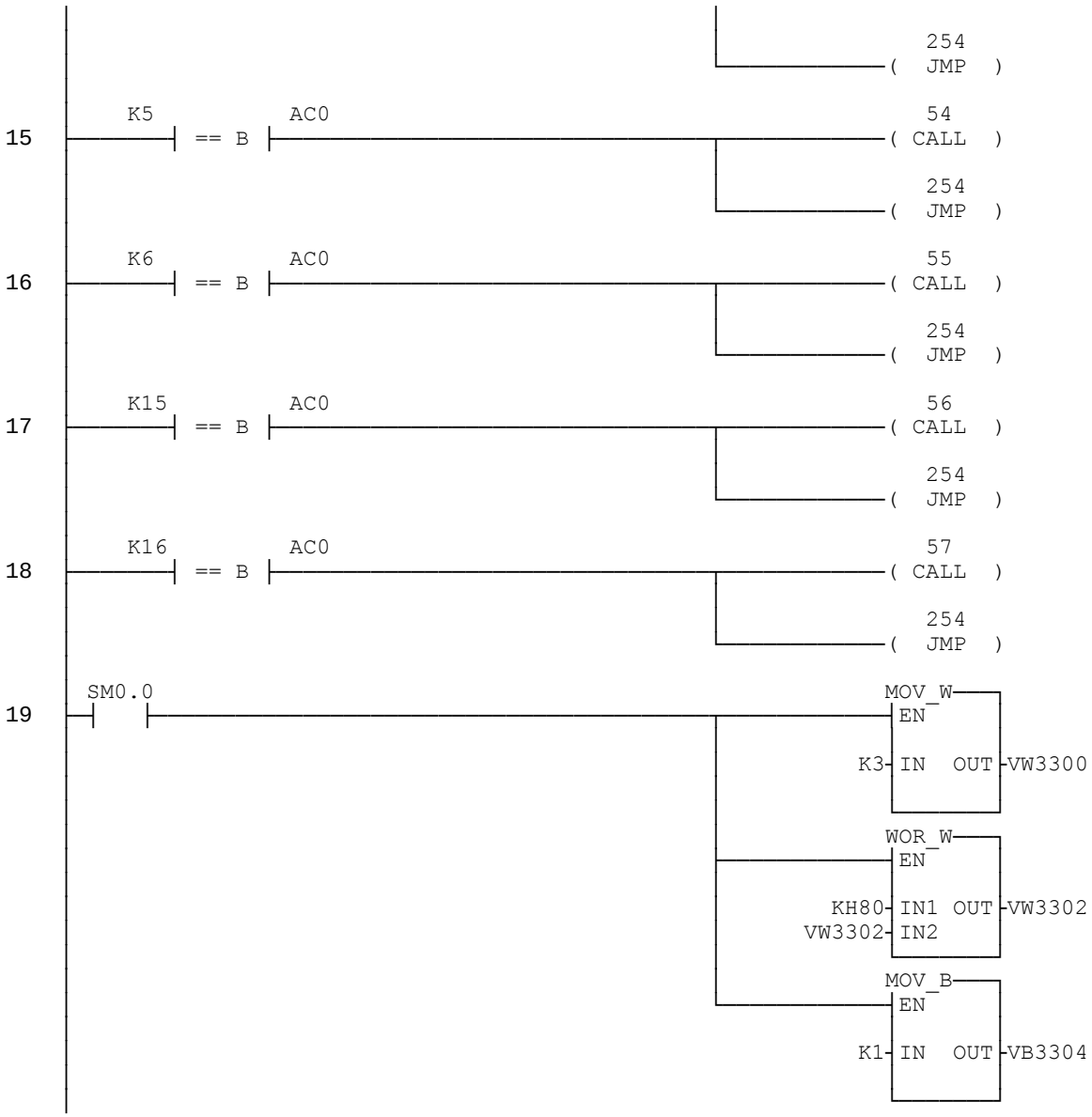
```
// Подпрограмма 51
//
// Эта подпрограмма обрабатывет запросы Modbus во время обычного цикла.
// Расчет CRC полученной телеграммы.  Если CRC, полученной телеграммы,
// включается в расчет, то результат всегда д.б. равен нулю, если
// не было ошибок.
```




```
SBR     51
LD      SM0.0
MOVW    VW3300,AC0      // получить длину буфера
MOVD    &VB3302,AC1     // получить адрес буфера для контроля CRC
CALL    63              // расчет CRC
LDW=    0,AC2           // Если (расчитанный CRC != 0),
NOT
JMP     255             // загрузить ошибку
```

```
// Телеграмма выглядит нормально, определяем какая функция Modbus затребована.
// Команда перехода, следующая за вызовом, будет всегда выполняться после обработки
// вызова, т.к. подпрограммы всегда устанавливают TOS в 1 перед возвратом.
```

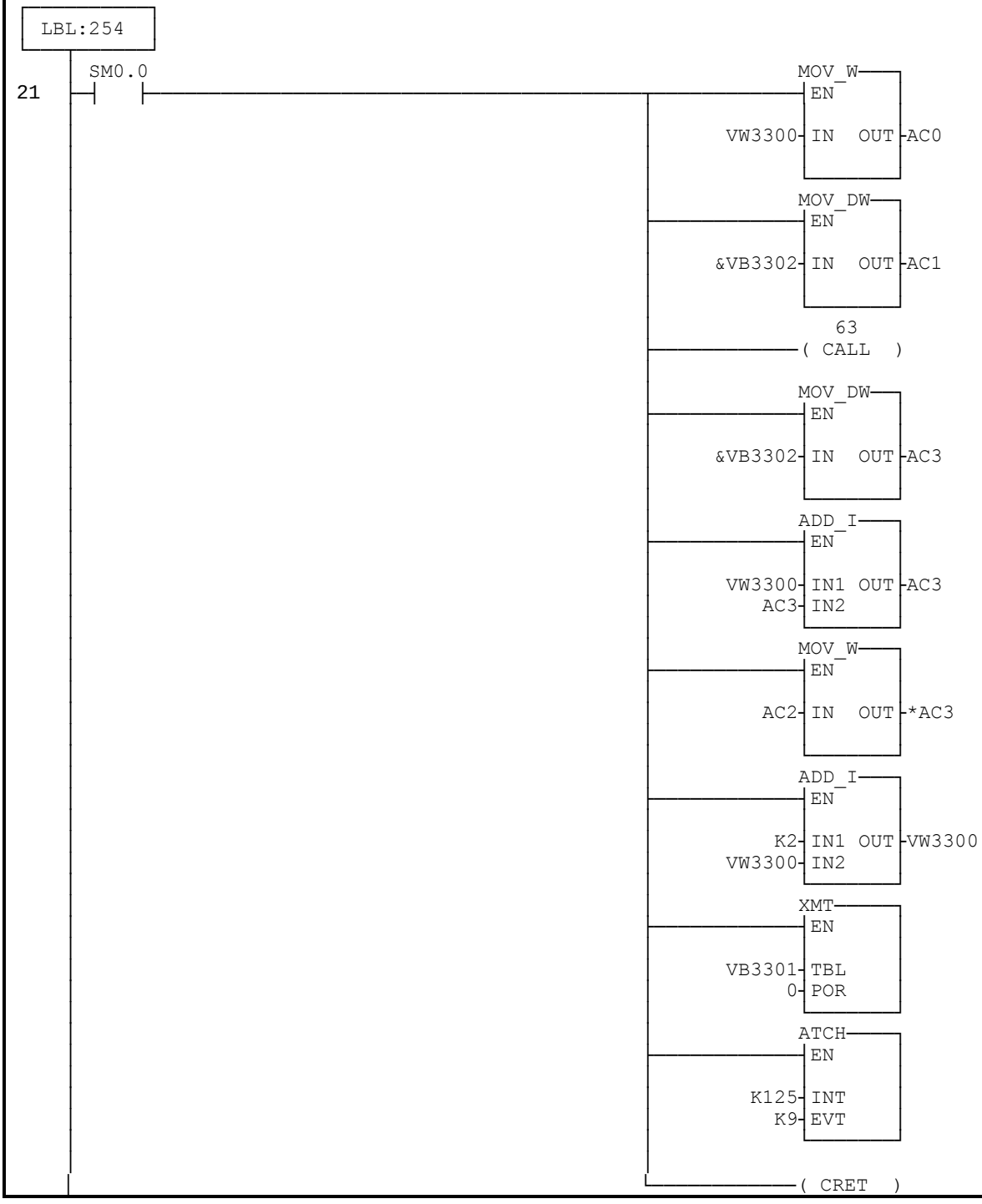





```
LD      SM0.0
MOVB    VB3303,AC0      // получить функцию из буфера приема

LDB=    1,AC0           // это функция 1?
CALL    52              // ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    2,AC0           // это функция 2?
CALL    52              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    3,AC0           // это функция 3?
CALL    53              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    4,AC0           // это функция 4
CALL    53              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    5,AC0           // это функция 5?
CALL    54              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    6,AC0           // это функция 6?
CALL    55              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец
```

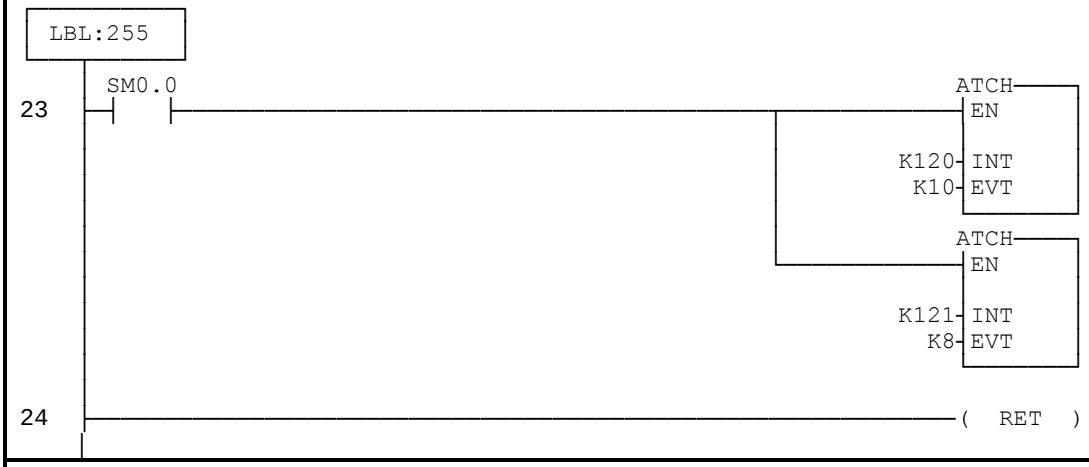
```
LDB=    15,AC0          // это функция 15?
CALL    56              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LDB=    16,AC0          // это функция 16?
CALL    57              // ... ...если да, обработать ее
JMP     254             //  ...затем перейти в конец

LD      SM0.0           // если ни одна из выше перечисленных ...
MOVW    3,VW3300        //   загрузить длину ответа об ошибке
ORW     16#0080,VW3302  //   установвить MSBit функции для показа ошибки
MOVB    1,VB3304        //   загрузить код "функция не поддерживается"
```

```
// Функция предназначена для расчета CRC ответа и  запуска передачи ответа.
// Подобно запросу длина ответа находиться в первом слове буфера.
// Эта длина не включает длину CRC и д.б. увеличена на два
// перед вызовом блока передачи.
```



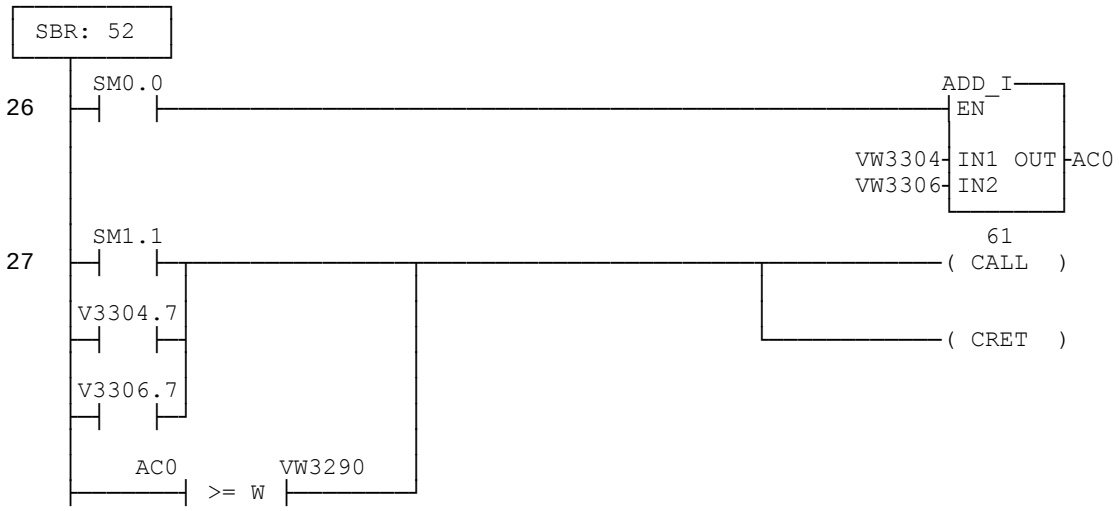
```
LBL     254
LD      SM0.0
MOVW    VW3300,AC0          // получить длину
MOVD    &VB3302,AC1         // получить адрес буфера для контроля CRC

CALL    63                  // рачет CRC

MOVD    &VB3302,AC3         // получить адрес начала буфера
+I      VW3300,AC3          // указатель на конец буфера
MOVW    AC2,*AC3            // поместить CRC в буфер
+I      2,VW3300            // добавить 2 байта к CRC

XMT     VB3301,0            // передача ответа
ATCH    125,9               // переход к INT 125, когда закончена xmit
CRET
```

```
// Обработка ошибок CRC или неверной длины.
//
// На самом деле не так много надо сделать в случае появления ошибки CRC
// или приема недостаточного для обработки количества байт, только
// сбросить драйвер для следующей телеграммы и вынудить активную
// станцию выдать тайм-аут.
```



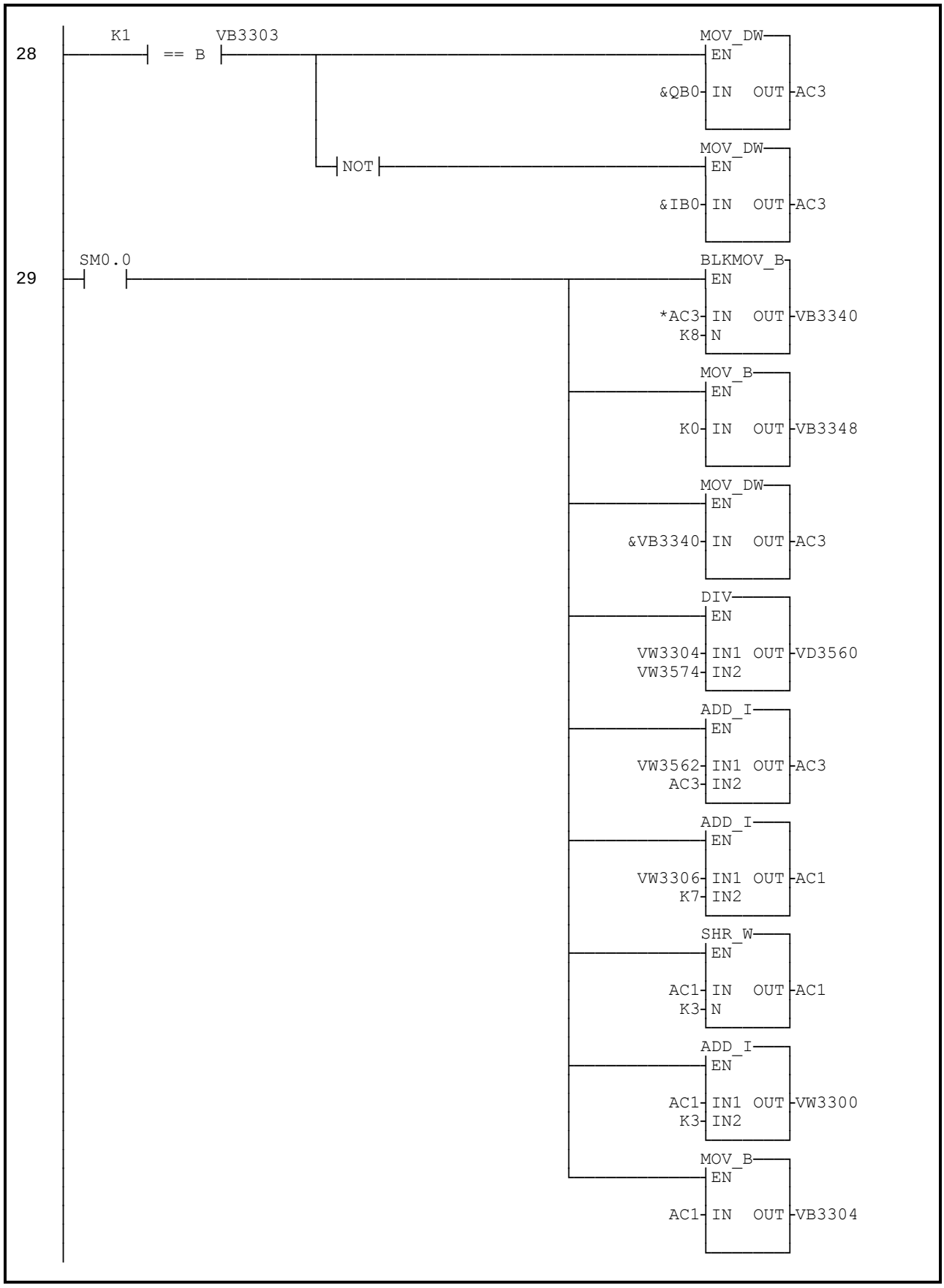

```
LBL     255
LD      SM0.0
ATCH    120,10              // запуск поиска for свободная линия
ATCH    121,8               // INT 121, если символ получен
RET
```

```
// Подпрограмма 52
//
// Эта подпраграмма поддерживает Modbus функции 1 и 2 - чтение состояния одного
// или нескольких выходов или входов. Биты в ответе собраны по восемь бит на байт.
// Первый запрошенный вход/выход находиться в младших битах (LSB) первого байта данных.
// Если число запршенных входов/выходов не делится на восемь, то старшие биты (MSB)
// последнего байта данных заполняются нулямиs, но это происходит не здесь.
//
// Формат запроса:
//   адрес   01   стартовый_бит (MSB,LSB)   количество_битов (MSB,LSB)
//
// стартовый_бит - первый запрошенный вход/выход (базис ноль).
// количество_битов - число запрошенных входов/выходов.
//
// Формат ответа:
//   адрес   01   количество_битов    данные.....
```



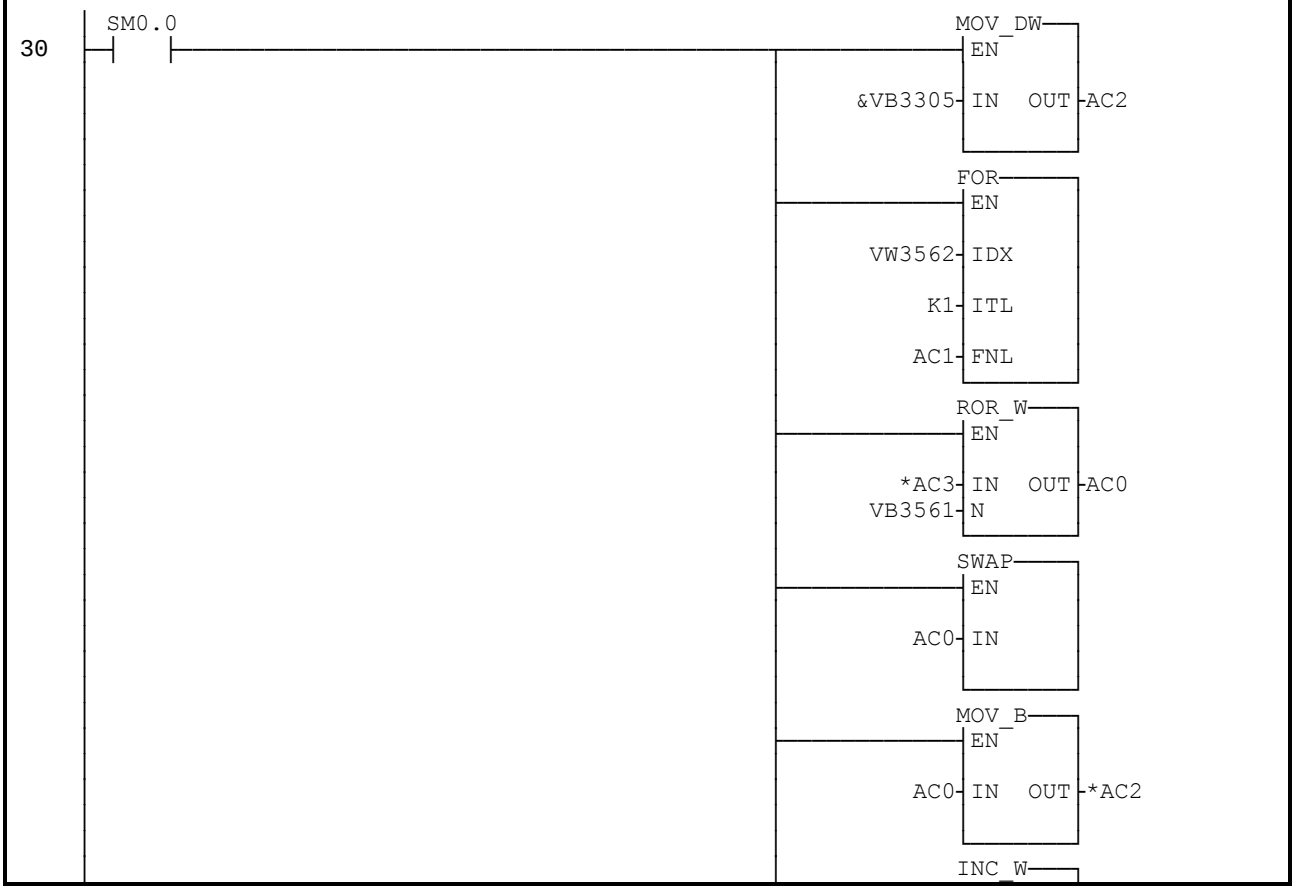

```
// Проверка, превышено ли максимальное количество входов или выходов.

SBR         52
LD          SM0.0
MOVW        VW3304,AC0              // получить значение стартового_бита
+I          VW3306,AC0              // ...и количество_битов

LD          SM1.1                   // Если (переполнение) или
O           V3304.7                 //   (стартовый_бит < 0) или
O           V3306.7                 //   (количество_битов < 0) или
OW>=        AC0,VW3290              //   (последний_номер_бита >
                                    //    макс._регистра_отображения)
CALL        61                      //   выдать ошибку
CRET                                //   возврат
```

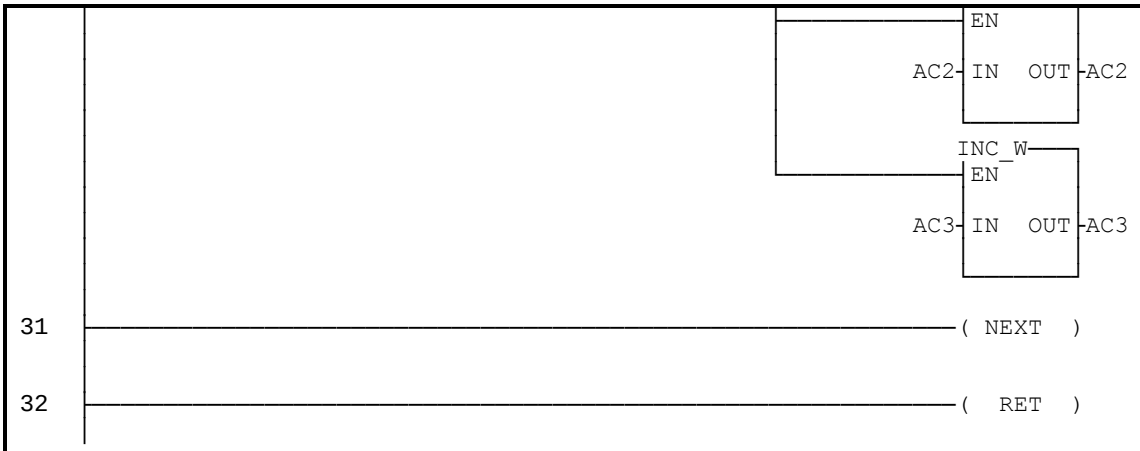
```
// Определение, с какой областью мы работаем, и затем копирование входов или
// выходов в рабочую область буфера обмена (не используемую в этом ответе),
// и установка последующих байтов в ноль. Это дает возможность правильно работать
// методу сдвига, используемому в следующей части.
//
// ЗАМЕЧАНИЕ:    Подразумевается, что размер регистра отображения - 8 байт.  Если размер
//               регистра отображения изменяется и больше 8 байт, то константа в команде
//               перемещения блока д.б. изменена.
//
// Получение номера стартового_бита и определение адреса первого байта для копирования.
// После деления значение в VW3560 - это остаток (количество сдвигов), а
// значение в VW3562 - это частное (смещение байта в входах или выходах).
```


28
K1
== B
VB3303
NOT
MOV_DW
EN
&QB0
IN
OUT
AC3
MOV_DW
EN
&IB0
IN
OUT
AC3
29
SM0.0
BLKMOV_B
EN
*AC3
IN
OUT
VB3340
K8
N
MOV_B
EN
K0
IN
OUT
VB3348
MOV_DW
EN
&VB3340
IN
OUT
AC3
DIV
EN
VW3304
IN1
OUT
VD3560
VW3574
IN2
ADD_I
EN
VW3562
IN1
OUT
AC3
AC3
IN2
ADD_I
EN
VW3306
IN1
OUT
AC1
K7
IN2
SHR_W
EN
AC1
IN
OUT
AC1
K3
N
ADD_I
EN
AC1
IN1
OUT
VW3300
K3
IN2
MOV_B
EN
AC1
IN
OUT
VB3304

```
LDB=    1,VB3303              // если функция 1 (выходы)
MOVD    &QB0,AC3              //    то указатель на выходы
NOT                           // иначе
MOVD    &IB0,AC3              //    указатель на входы

LD      SM0.0
BMB     *AC3,VB3340,8         // копирование входов/выходов в рабочую область
MOVB    0,VB3348              // обнуление "остальных" байтов в рабочей области
MOVD    &VB3340,AC3           // исходный указатель = адрес рабочей области

MOVW    VW3304,VW3562         // получить значение стартового_бита
DIV     VW3574,VD3560         // стартовый_бит / 8
+I      VW3562,AC3            // добавить смещение к исходному указателю

// Получение количества_битов и определение числа байтов, необходимых для передачи
// ответа.

MOVW    VW3306,AC1            // получить количество_битов
+I      7,AC1                 // округлить
SRW     AC1,3                 // разделить на 8 бит/байт

// Загрузить размер буфера ответа и количество_байт в буфер ответа.

MOVW    AC1,VW3300            // получить количество_байтов
+I      3,VW3300              // добавить три служебных байта
MOVB    AC1,VB3304            // загрузить количество_байтов
```

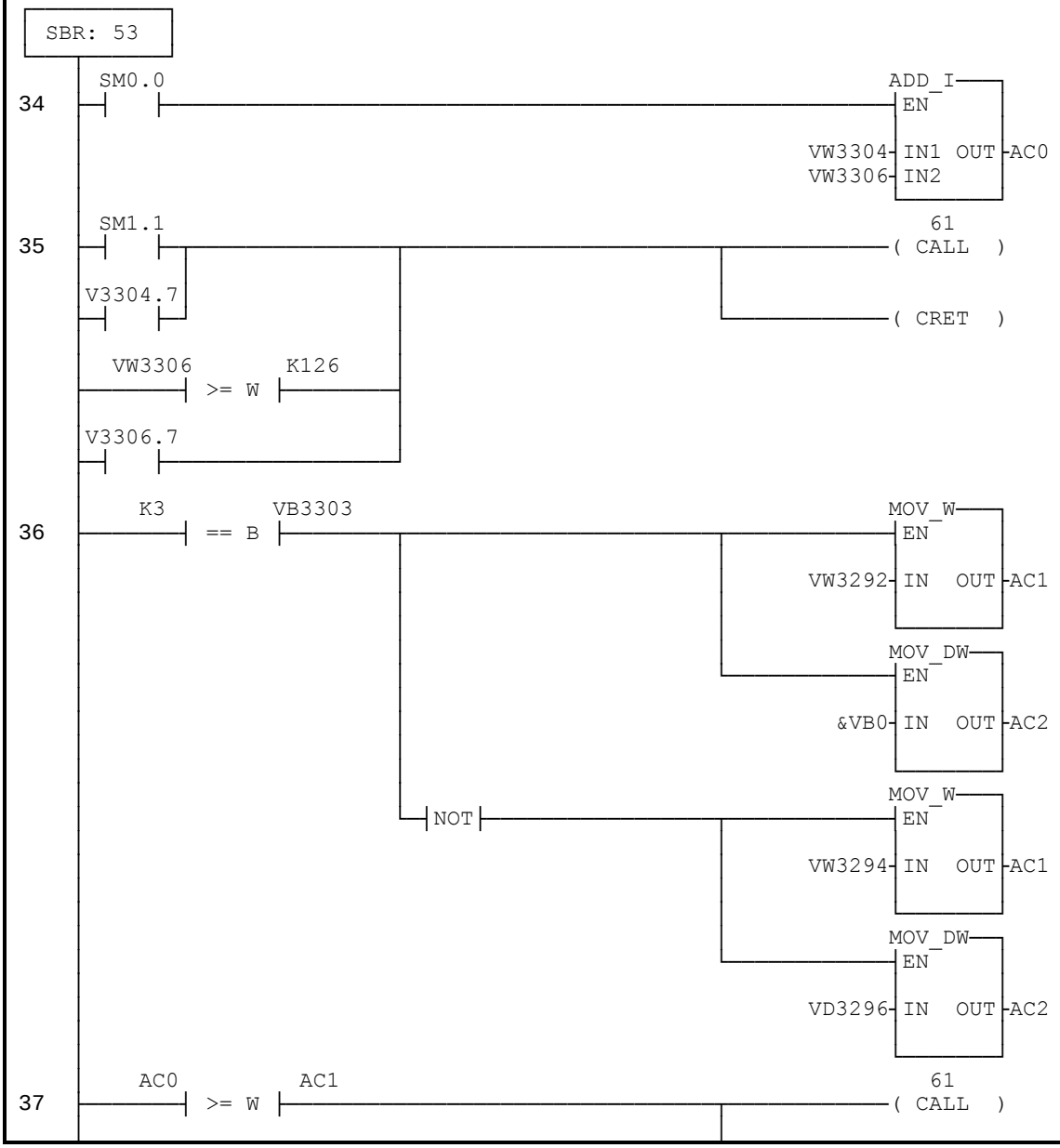
```
// Получение данных из регистра отображения, сдвиг и помещение их в буфер ответа.
//
// Используемый метод: в старший байт слова выбирается слово с необходимым байтом
// из регистра отображения. Слово циклически сдвигается на число бит.
// Таким образом, младший бит "следующего" байта регистра отображения сдвигается
// в старший бит нужного байта. Когда сдвиг закончен, то с помощью команды обмена,
// мы получаем байт в младшем байте слова и затем копируем байт в буфер ответа.
```






```
LD      SM0.0
MOVD    &VB3305,AC2          // загрузить буфер ответа

FOR     VW3562,1,AC1         // для количества_байт байты
MOVW    *AC3,AC0             //   получить слово выходов
RRW     AC0,VB3561           //   сдвиг его вправо
SWAP    AC0                  //   поместить байт в младший байт
MOVB    AC0,*AC2             //   запомнить байт
INCW    AC2                  //   инкрементировать указатель буфера
INCW    AC3                  //   инкрементировать указатель регистра отображения
NEXT                         // конец для
RET
```

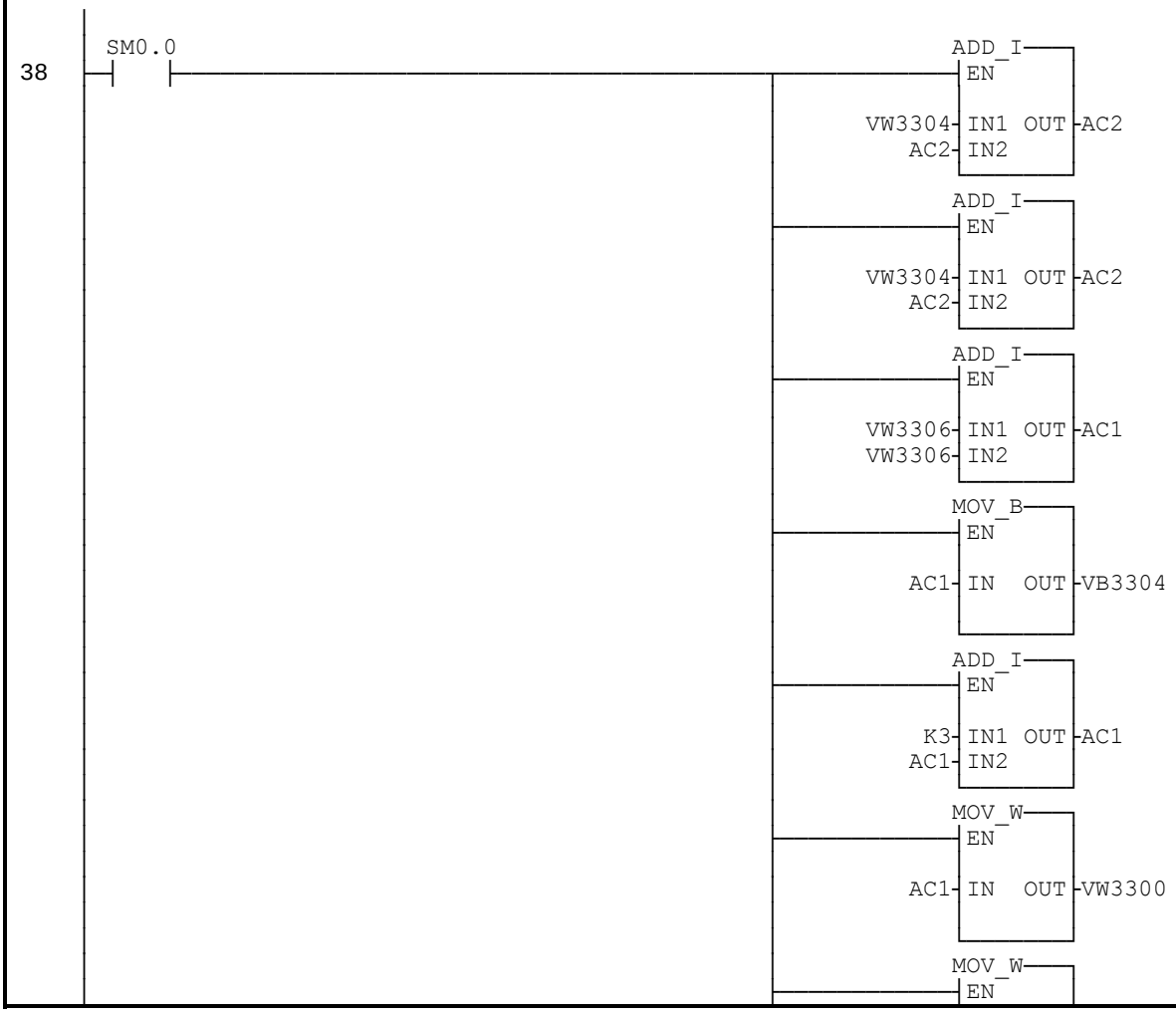
```
// Подпрограмма 53
//
// Эта подпрограмма поддерживает Modbus функцию 3 (чтение регистров выходов /
// управляющих регистров)и функцию 4 (чтение регистров входов). Подразумевается, что
// регистры выходов / управляющие регистры находяться в V памяти в ПЛК.
// Регистры входов - аналоговые входы.
//
// ЗАМЕЧАНИЕ: аналоговые входы не могут напрямую быть прочитаны в данной подпрограмме,
//                  т.к. ПЛК не разрешает косвенный доступ к аналоговым значениям. Аналоговые
//                  значения читаются из области V памяти, заданной в VD3296. Эта область д.б.
//                  задана в подпрограмме инициализации. Копирование аналоговых данных в
//                  V память - это обязанность пользователя.
//
// Формат для запроса:
//
//   адрес   03   начальное_слово (MSB,LSB) количество_слов(MSB,LSB)
//
//  начальное_слово - первое запршенное слово (базис ноль).
//  количество_слов - число запрошенных слов.
//
// Формат для ответа:
//
//   адрес   03   количество_байт   данные.....
```

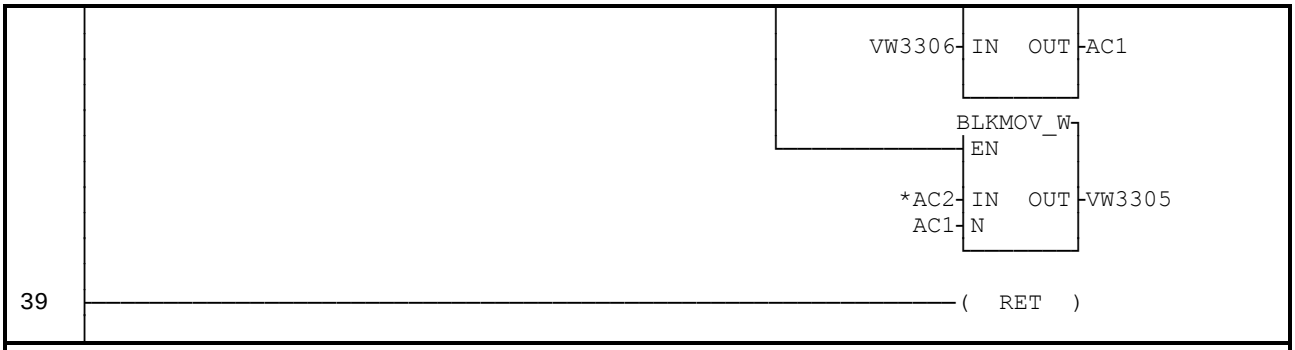


```
                                                    ( CRET   )
```

```
// Проверка, превышена ли макс. граница памяти.

SBR         53
LD          SM0.0
MOVW        VW3304,AC0          // получить значение начального_слова
+I          VW3306,AC0          // ...плюс   количество_слов

LD          SM1.1               // если (переполнение) или
O           V3304.7             //   (начальное_слово < 0) или
OW>=        VW3306,126          //   (количество_слов > 125) или
O           V3306.7             //   (количество_слов < 0)
CALL        61                  //   выдать ошибку отклика
CRET                            //   возврат

LDB=        3,VB3303            // если функция 3 (V память)
MOVW        VW3292,AC1          //   получить макс. размер V памяти
MOVD        &VB0,AC2            //   исходный_указатель = V память
NOT                             // иначе
MOVW        VW3294,AC1          //   получить макс. размер слов AI
MOVD        VD3296,AC2          // исходный_указатель = AI память

LDW>=   AC0,AC1                 // если начальное_слово + количество_слов >=
                                // размера памяти
CALL        61                  //   выдать ошибку
CRET                            //   возврат
```

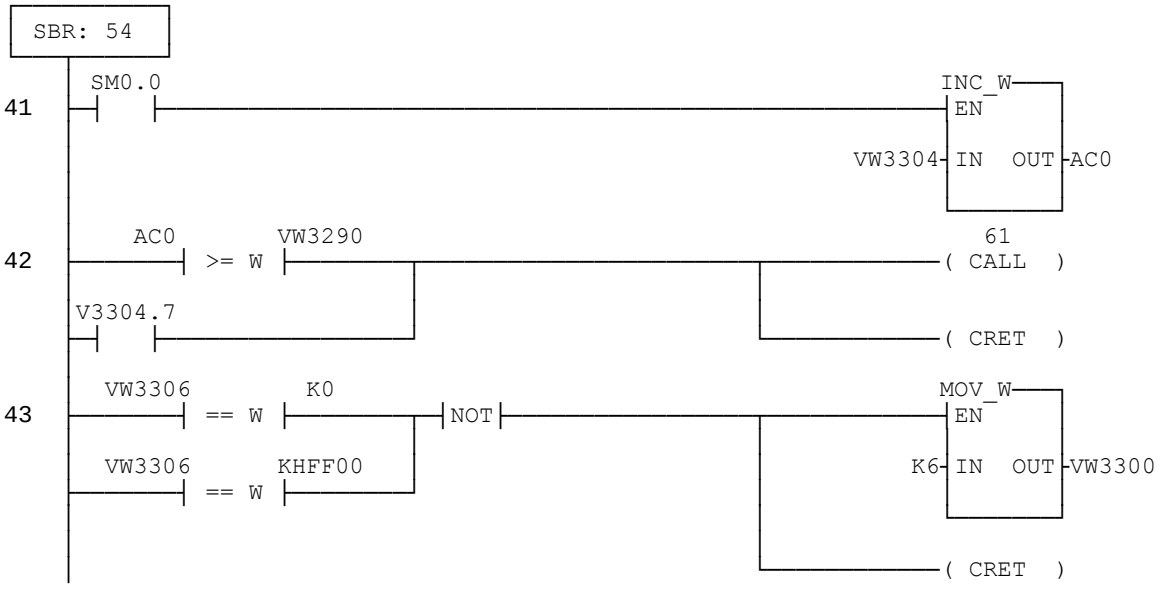
```
// Запрос - OK; получить начальное_слово, удвоить его для смещения байта и
// добавить его к исходному_указателю. Указатель на буфер ответа и
// скопировать данные сюда.
```





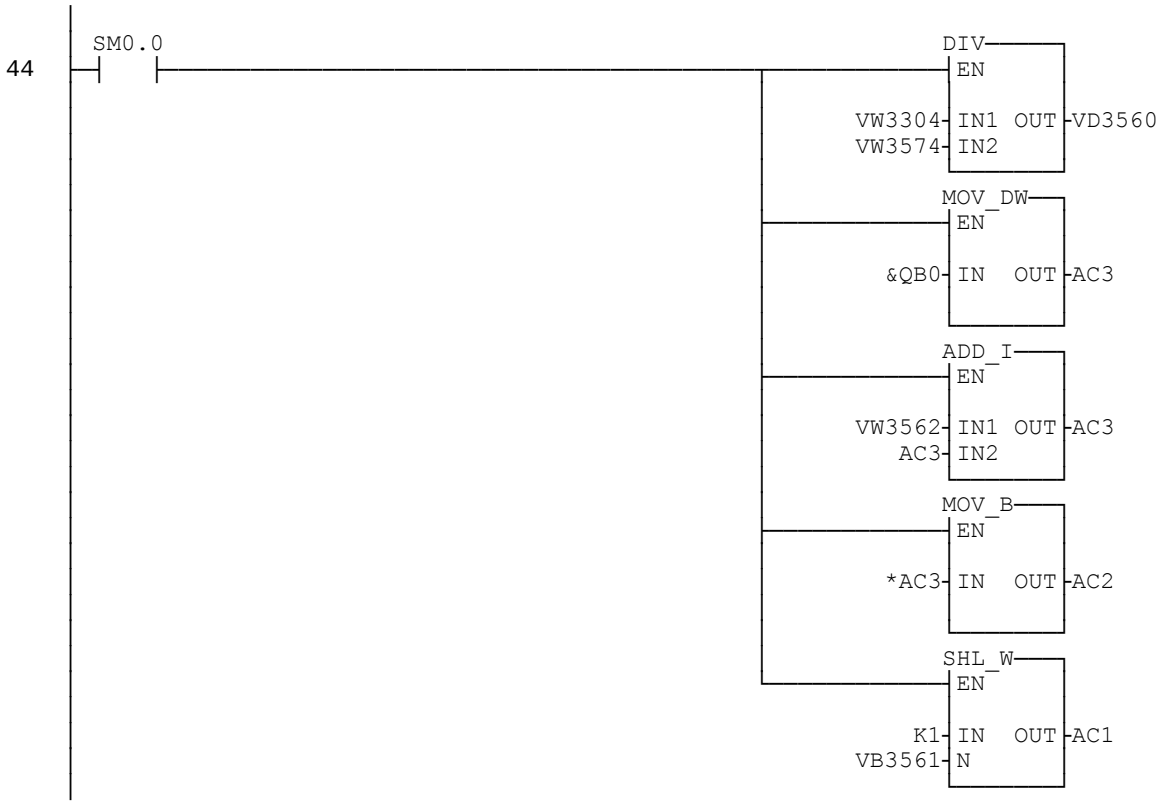

```
LD      SM0.0
+I      VW3304,AC2      // добавить начальное_слово к исходному_указателю
+I      VW3304,AC2      //  ...начальное_слово * 2 к исходному_указателю

MOVW    VW3306,AC1      // получить количество слов
+I      VW3306,AC1      // ...умножить на два для количества байт
MOVB    AC1,VB3304      //  ...запомнить количество байт ответа
+I      3,AC1           // добавить служебные байты
MOVW    AC1,VW3300      //  ...запомнить размер буфера ответа

MOVW    VW3306,AC1      // получить опять количество слов
BMW     *AC2,VW3305,AC1 // скопировать данные из источника в буфер
RET                     // возврат
```

```
// Подпрограмма 54
//
// Даная подпрограмма поддерживает Modbus функцию 5 для управления отдельным выходом: // включения
или выключения.
//
// Формат запроса:
//
//   адрес   05   бит_входов (MSB,LSB)   данные (FF00 или 0000)
//
// Значение данных FF00 включает вход, значение 0000 выключает вход.
// Любое другое значение данных не приводит ни к какому действию.
//
// Ответная телеграмма повторяет телеграмму запроса.
```

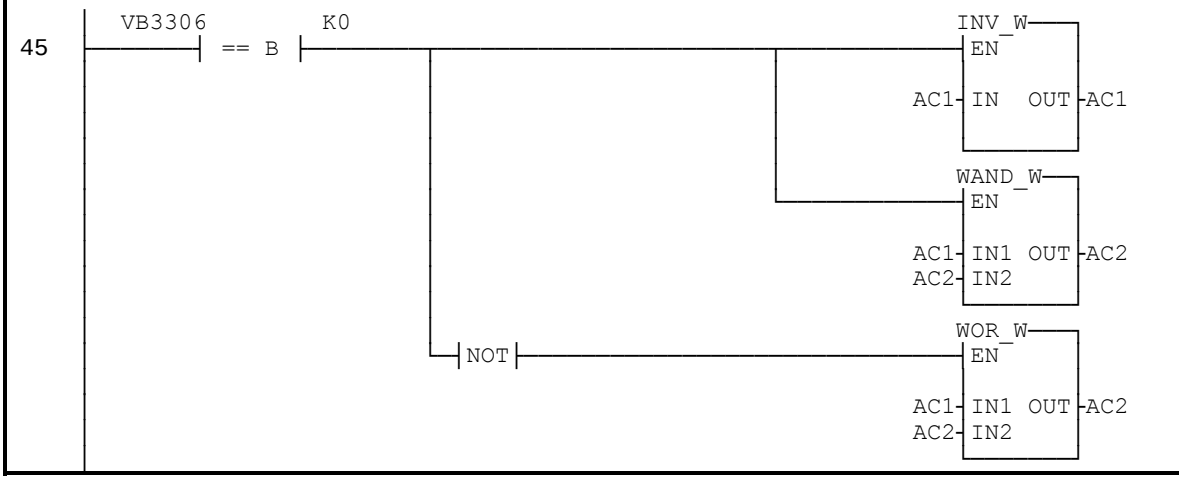



```
SBR         54
LD          SM0.0
MOVW        VW3304,AC0              // получить бит_выходов
INCW        AC0                     // ...и запомнить его для проверки

LDW>=   AC0,VW3290                  // если (бит_выходов >
                                    // макс_число_регистра_отображения) или
O           V3304.7                 //   (бит_выходов < 0)
CALL        61                      //   выдать ошибку
CRET                                //   возврат

// Проверка, равны ли данные FF00 или 0000.  Если данные не совпадают ни с одним
// из этих значений, то не предпринимаются никакие действия, за исключением
// повторения запроса к активной станции.

LDW=        VW3306,0                // если (данные != 0) или
OW=         VW3306,16#FF00          //   (данные != FF00)
NOT
MOVW        6,VW3300                //   установить длину ответа
CRET                                //   возврат
```

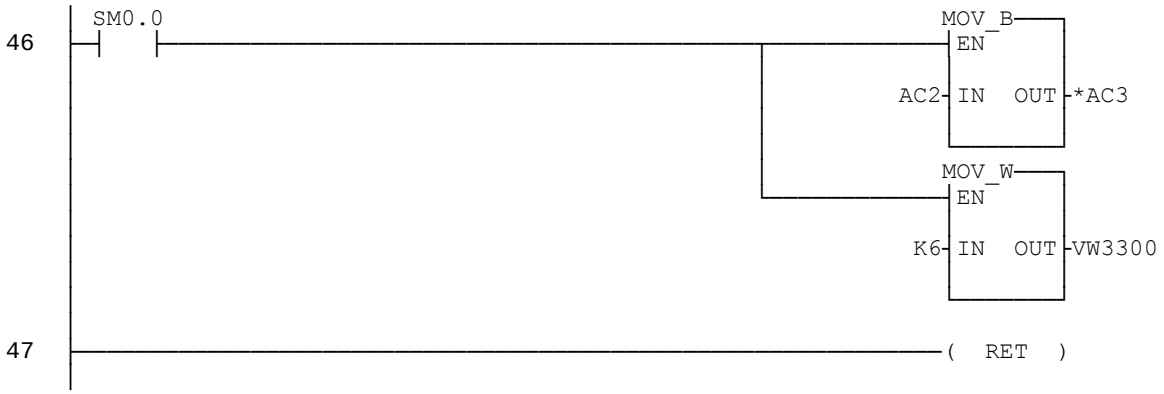
```
// Определение какой байт выходов и какой бит байта изменились.
// Определяется при помощи деления бита выходов на восемь. Частное - байт
// смещения, а остаток - номер бита.
//
// После деления, значение в VW3560 - остаток (номер бита), а
// значение в VW3562 - частное (смещение в выходах).
```




```
LD      SM0.0
MOVW    VW3304,VW3562       // загрузить бит_выходов
DIV     VW3574,VD3560       // бит_выходов / 8

MOVD    &QB0,AC3            // указатель на выходы
+I      VW3562,AC3          // сместить указатель на соответствующий байт
MOVB    *AC3,AC2            // получить байт выходов

MOVW    1,AC1               // создать маску
SLW     AC1,VB3561          // ...для соответствующего бита
```

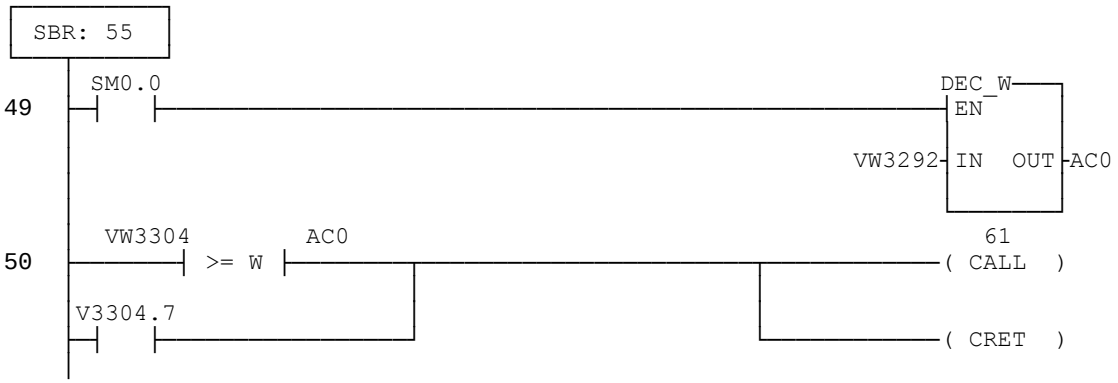


```
LDB=    VB3306,0        // если данные == 0
INVW    AC1             //   образовать маску очистки
ANDW    AC1,AC2         //   очистить бит
NOT                     // иначе
ORW     AC1,AC2         //   установить бит
```



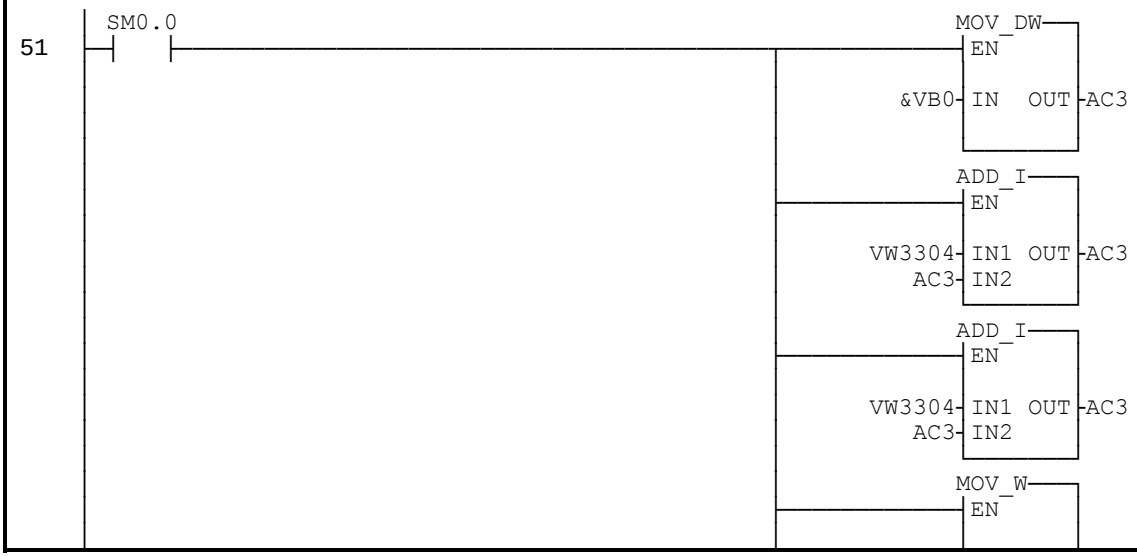

```
LD      SM0.0
MOVB    AC2,*AC3        // восстановить байт выхода
MOVW    6,VW3300        // установить длину ответа
RET
```

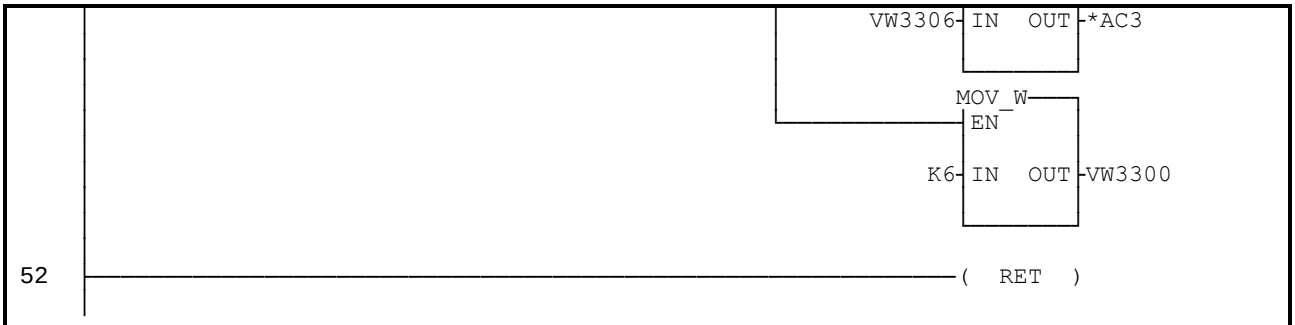
```
// Подпрограмма 55
//
// Данная подпрограмма поддерживает Modbus функцию 6 для записи одного управляющего
// регистра в ПЛК.  Управляющие регистры для данной реализации подразумеваются в
// V памяти.
//
// Формат запроса:
//
//   адрес   06   начальное_слово (MSB,LSB)   данные (MSB,LSB)
//
//  начальное_слово - первое запрошенное слово (базис ноль).
//  данные - слово, записываемое в ПЛК.
//
// Формат ответа такой же, как запроса:
//
//   адрес   06   начальное_слово (MSB,LSB)   данные (MSB,LSB)
```




```
// Проверка, не превышен ли макс. предел памяти.

SBR         55
LD          SM0.0
MOVW        VW3292,AC0          // получить значение размера памяти + 1
DECW        AC0                 // ...и получить из него размер памяти

LDW>=   VW3304,AC0              // если (начальное_слово >= размер памяти) или
O           V3304.7             //   (начальное_слово < 0)
CALL        61                  //   выдать ошибку
CRET                            //   возврат
```

```
// Запрос - OK; получить начальное_слово, удвоить его для смещения байта, и
// добавить его к исходному_указателю. Скопировать данные запроса в V память.
```

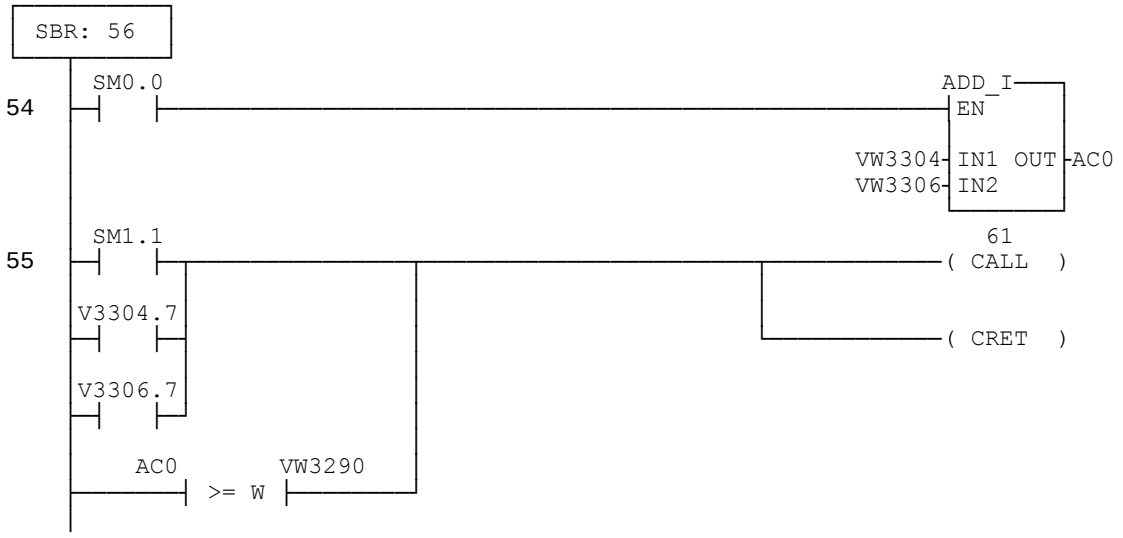





```
LD      SM0.0
MOVD    &VB0,AC3        // исходный_указатель = V память
+I      VW3304,AC3      // добавить смещение...
+I      VW3304,AC3      // ...дважды для смещения байта
MOVW    VW3306,*AC3     // запомнить данные в V память
MOVW    6,VW3300        // установить длину ответа
RET                     // возврат
```

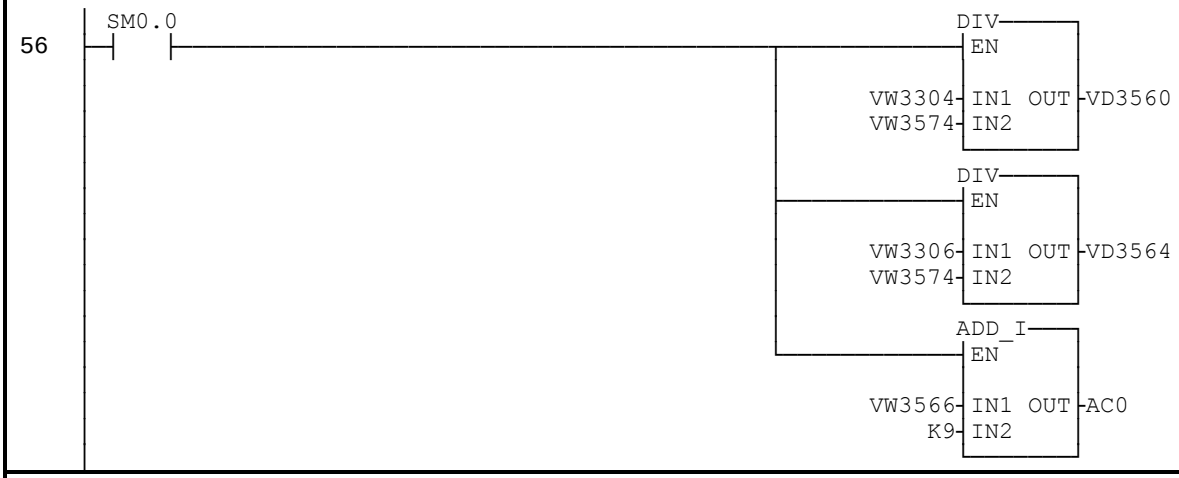
```
// Подпрограмма 56
//
// Данная подпрограмма поддерживает Modbus функцию 15 (управление несколькими
// выходами). Выхода имеют восемь бит на байт. LSБит первого байта данных
// записывается в ячейку начального_бита. Последующие биты записываются
// в следующие ячейки.
//
```

```
// ЗАМЕЧАНИЕ: В этой реализации, если начальный_бит и значение количества_бит
//                  не кратны восьми, возвращается ошибка.  В STL или LAD очень трудно
//                  обрабатывать начало любого бита.
//
//  Формат запроса:
//
//  адрес  0F  начальный_бит(MSB,LSB)  количество_бит(MSB,LSB)  количество_байт  данные...
//
//  начальный_бит - первый записываемый выход и LSБит первого байта данных.
//  количество_бит - число записываемых выходов.
//  количество_бит - число байтов данных.
//
//  Формат ответа:
//
//  адрес  0F  начальный_бит(MSB,LSB)  количество_бит(MSB,LSB)
```



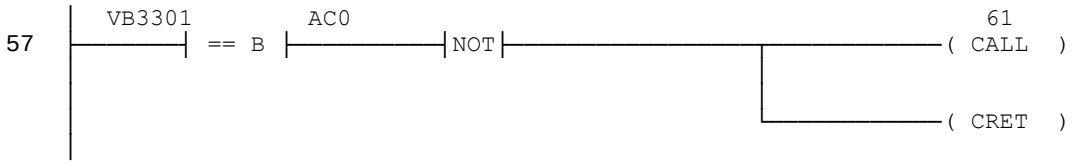

```
// Проверка, превышено ли макс. число выходов.

SBR         56
LD          SM0.0
MOVW        VW3304,AC0          // получить значение начального_бита
+I          VW3306,AC0          // ...плюс количество_бит

LD          SM1.1               // если (переполнение) или
O           V3304.7             //   (начальный_бит < 0) или
O           V3306.7             //   (количество_бит < 0) или
OW>=        AC0,VW3290          //   (последний_адрес > макс_регистра_отображения)
CALL        61                  //   выдать ошибку
CRET                            //   возврат
```

```
// Определение смещения байта и бита в регистре отображения выхода, и полная или
// частичная запись байтов.
//
// После деления, значение в VW3560 - остаток (количество сдвигов), а
// значение в VW3562 - частное (байт смещения в выходах).
//
// После деления, значение в VW3564 - остаток, а значение в
// VW3566 - число полных записанных байт.
```



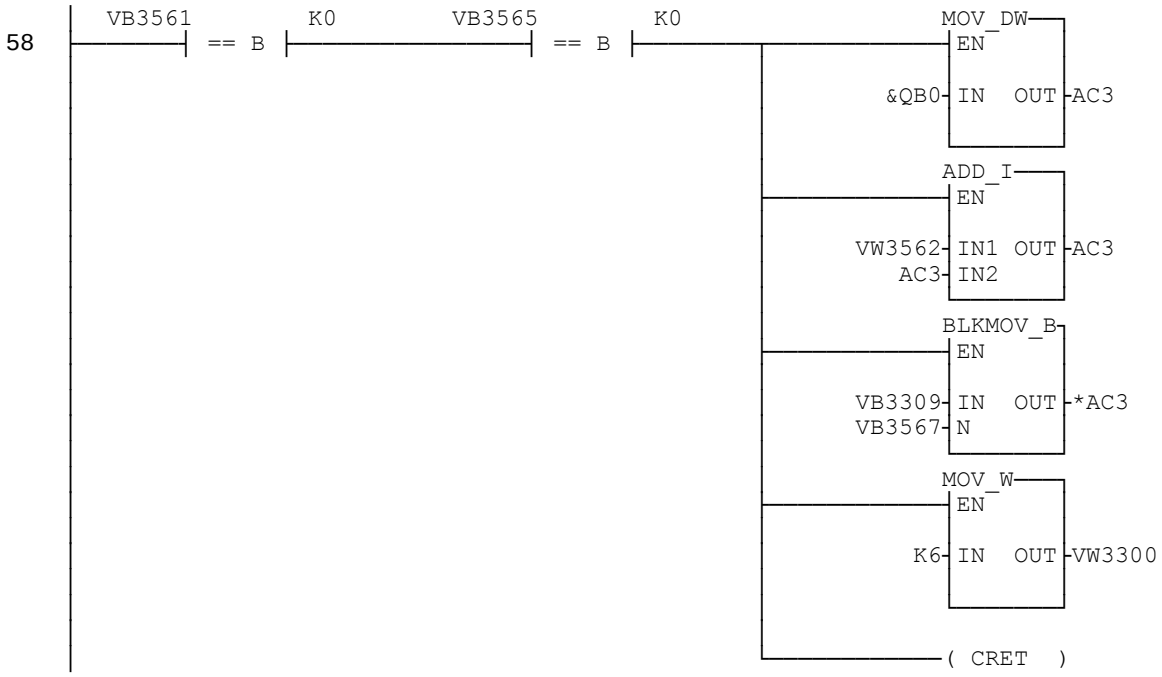

```
LD      SM0.0
MOVW    VW3304,VW3562       // получить значение начального_бита
DIV     VW3574,VD3560       // начальный_бит / 8

MOVW    VW3306,VW3566       // получить количество_бит
DIV     VW3574,VD3564       // количество_бит / 8
MOVW    VW3566,AC0          // получить (количество_бит / 8)
+I      9,AC0               // ...плюс 9 служебных байт
```

```
// Будте уверены, что главная ЭВМ посылает достаточно данных для запрос.
// Размер буфера д.б. равен (количеству_бит / 8) + 9 служебных байт.
```



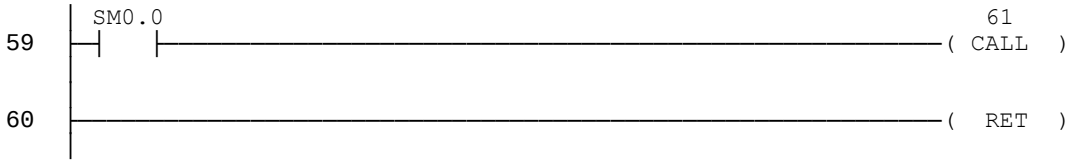

```
LDB=    VB3301,AC0          // если (расчитанная длина ( полученной длине)
NOT
CALL    61                  //   выдать ошибку
CRET                        //   возврат
```

```
// Определение работаем ли мы с "удобным" кадром, где начальный_бит это
// первый бит байта и количество_бит - целое число байт.
// Если это не так, то выдать ошибку главной ЭВМ.
```



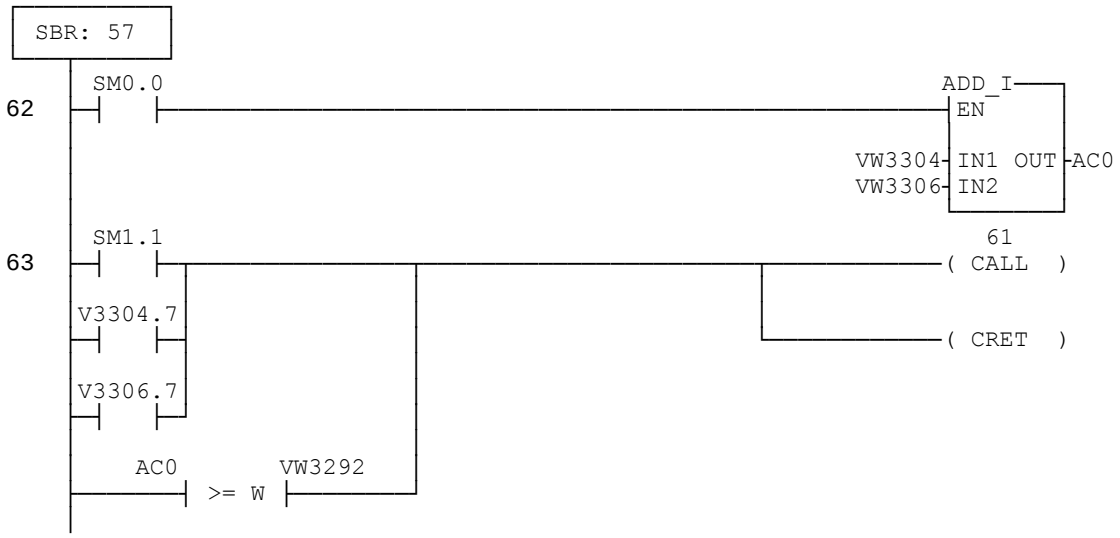

```
LDB=    VB3561,0              // если (начальное_бит - нулевой бит байта) и
AB=     VB3565,0              //   (целое число байт = TRUE)
MOVD    &QB0,AC3              //   указатель на выходы
+I      VW3562,AC3            //   добавить смещение к начальному выходу
BMB     VB3309,*AC3,VB3567    //   скопировать запрашиваемые данные на выхода
MOVW    6,VW3300              //   установить размер ответа
CRET                          //   возврат
```

```
// Это не удобный кадр; выход с прерыванием запроса.
```



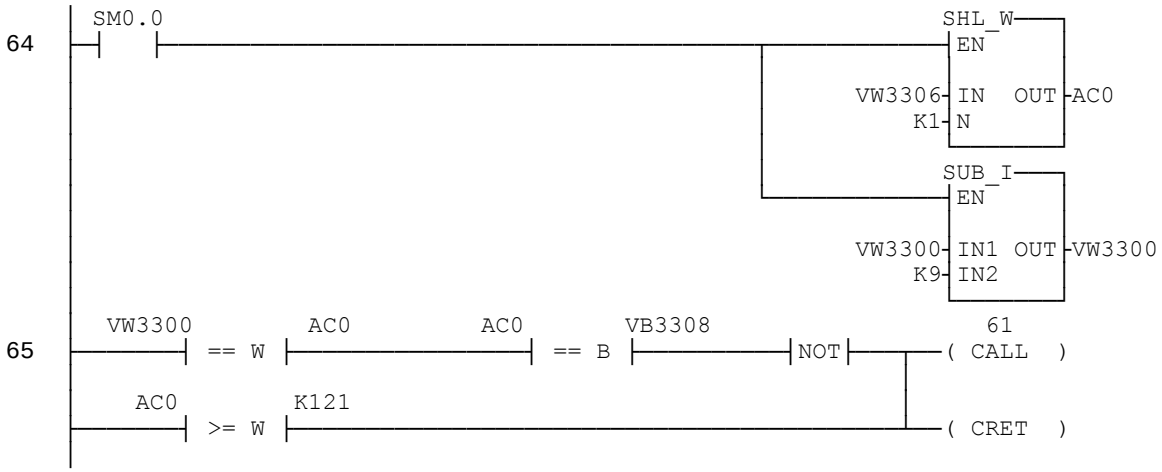

```
LD      SM0.0                 // иначе
CALL    61                    //   выдать ошибку
RET                           //   возврат
```

```
// Подпрограмма 57
//
// Данная подпрограмма поддерживает Modbus функцию 16 (запись до 60 управляющих
// регистров в ПЛК). В данной реализации подразумевается, что управляющие регистры
// находяться в V памяти.
//
// ЗАМЕЧАНИЕ: макс. значение 60 регистров (120 байт) - ограничение записанное в
//                      спецификации Modbus.
//
//  Формат запроса:
//   адрес  10  начальное_слово(MSB,LSB) количество_слов(MSB,LSB) количество_байт  данные...
//
//  Формат ответа такой же, как запроса:
//  адрес  10  начальное_слово(MSB,LSB) количество_слов(MSB,LSB)
```



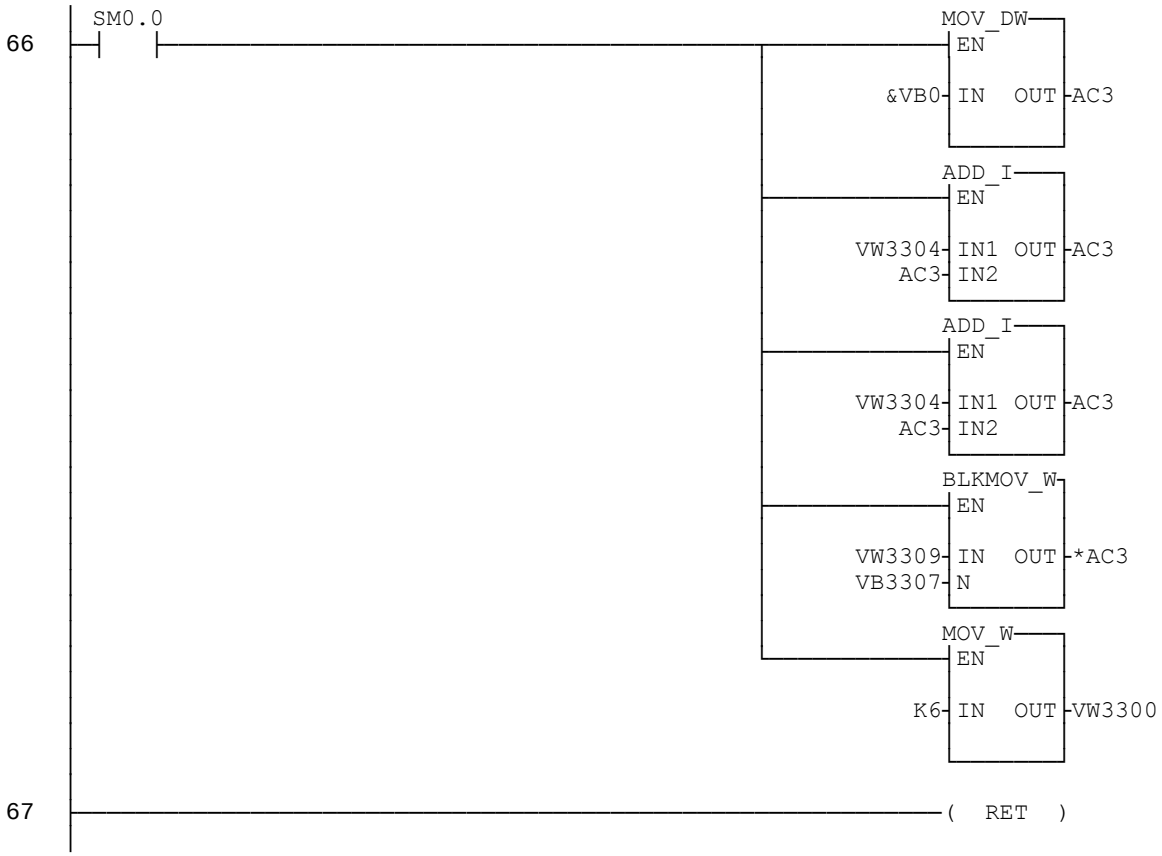

```
// Расчет адреса последнего запршенного слова путем добавления количества_слов к
// начальному_слову. Если этот последний адрес больше, чем макс. адрес доступной V памяти,
// выдать ошибку.

SBR       57
LD        SM0.0
MOVW      VW3304,AC0          // получить начальное_слово
+I        VW3306,AC0          // ...плюс количество_слов

LD        SM1.1               // если (переполнение) или
O         V3304.7             //   (начальное_слово < 0) или
O         V3306.7             //   (количество_слов < 0) или
OW>=      AC0,VW3292          //   (последний адрес >= размер памяти)
CALL      61                  //   выдать ошибку
CRET                          //   возврат
```

```
// Соравнение количества принятых байт с количеством_слов телеграммы, чтобы
// быть уверенным, что мы получили достаточно байт данных. Принятое количество_байт
// д.б. равно (количество_слов * 2) + 9 служебных байт. Так же проверка, что
// количество слово не больше 60 (120 байт), макс. разрешенное для функции.
//
// Так же проверка, что количество_слов * 2 равно количеству_байт. Двойная проверка всех
// этих полей - это небольшая избыточность, но мы должны выполнить все, что положено.
```



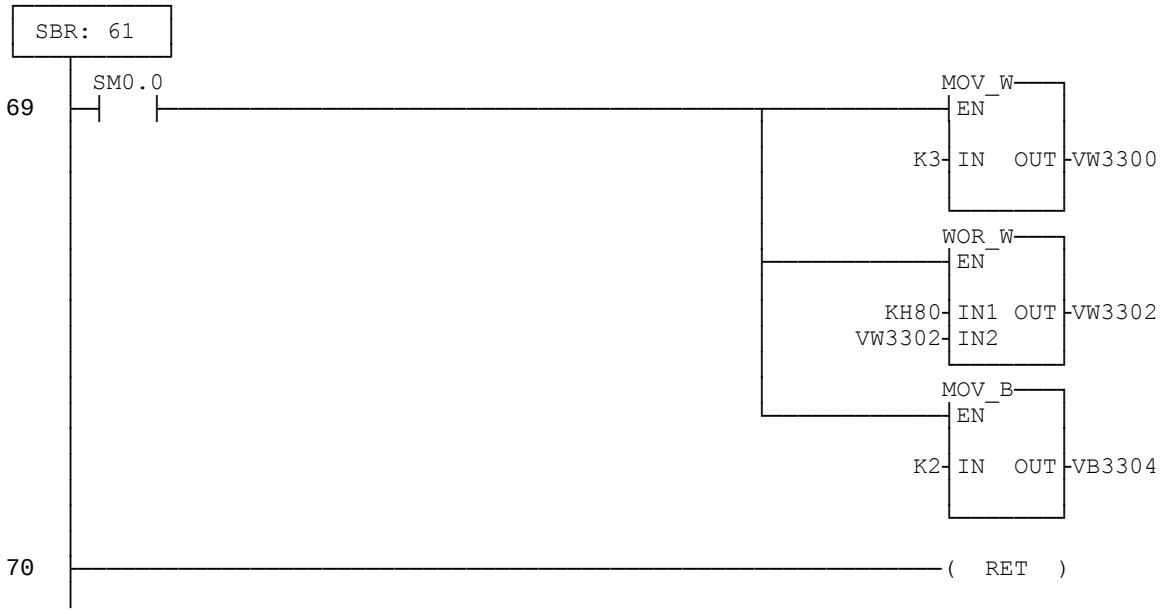

```
LD          SM0.0
MOVW        VW3306,AC0          // получить количество_слов
SLW         AC0,1               // количество_слов * 2
-I          9,VW3300            // количество_байт - 9

LDW=        VW3300,AC0          // если (количество_слов * 2 !=
                                // принятому количеству - 9) или
AB=         AC0,VB3308          //   (количество_слов * 2 != количество_байт) или
NOT
OW>=   AC0,121                  //   (количество > 60 слов + служебные)
CALL        61                  //   выдать ошибку
CRET                            //   возврат
```

```
// Запрос - OK; получить начальное_слово, удвоить его для смещения байта, а затем
// добавить его к исходному_указателю. Скопировать данные запрос в V память.
```



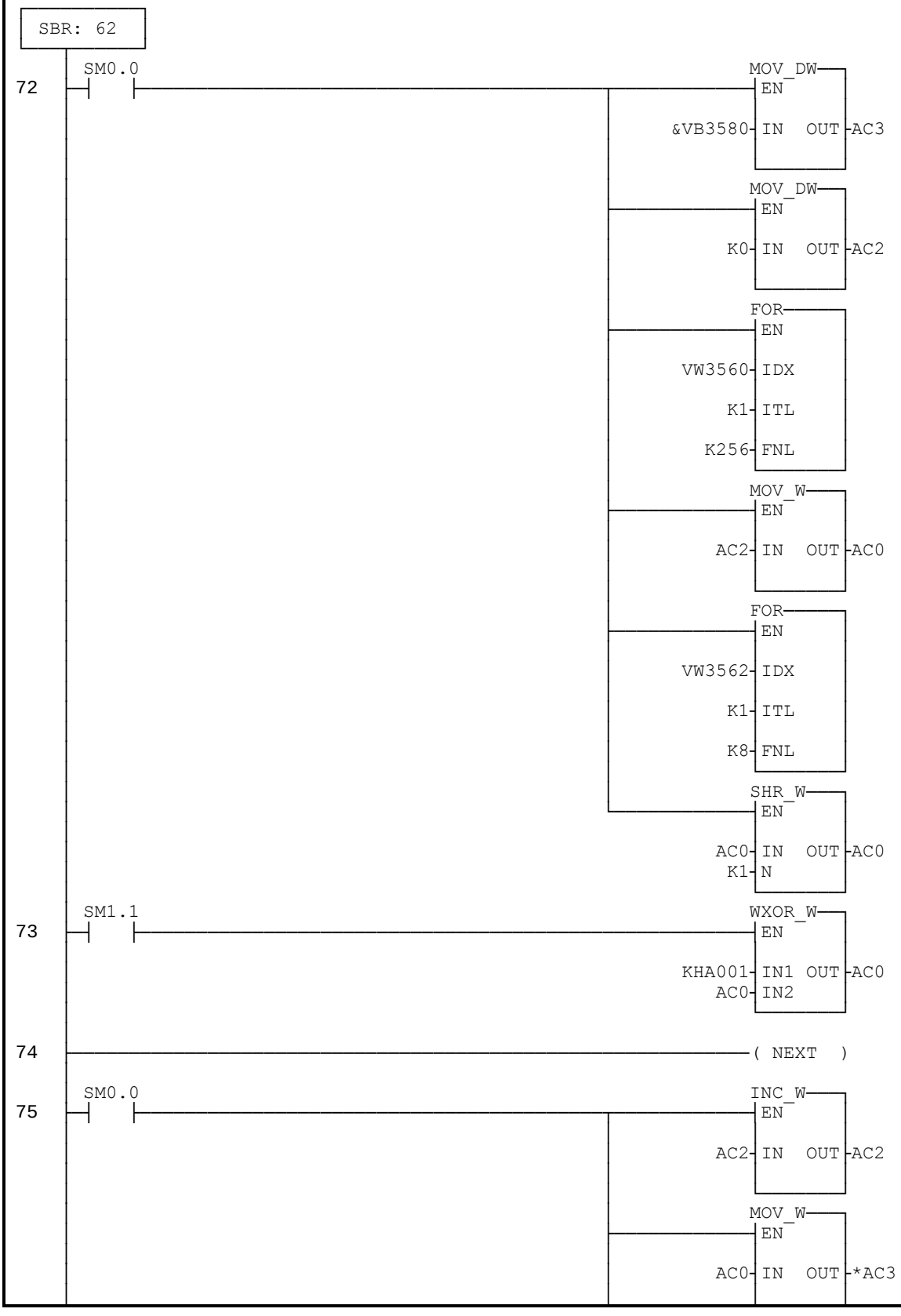

```
LD      SM0.0
MOVD    &VB0,AC3              // исходный_указатель = V память
+I      VW3304,AC3            // добавить смещение...
+I      VW3304,AC3            // ...дважды для смещения байта

BMW     VW3309,*AC3,VB3307    // запомнить данные в V память
MOVW    6,VW3300              // установить длину ответа
RET                           // возврат
```

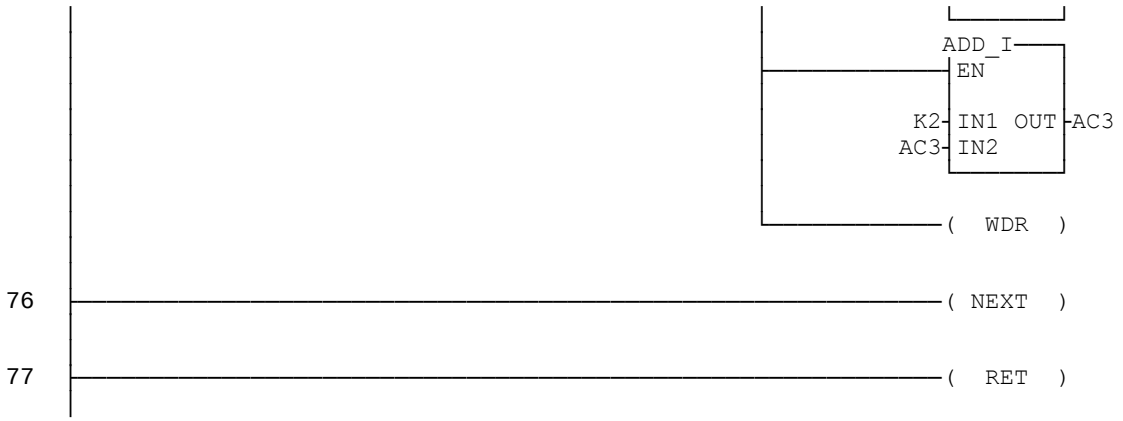
```
// Подпрограмма 61
//
// Данная подпрограмма устанавливает буфер ответа для особого кода Modbus два.
```




```
SBR     61
LD      SM0.0
MOVW    3,VW3300          // загрузить длину ответа об ошибке
ORW     16#0080,VW3302    // установить MS-Бит функции для показа ошибки
MOVB    2,VB3304          // загрузить код особой ситуации

RET
```

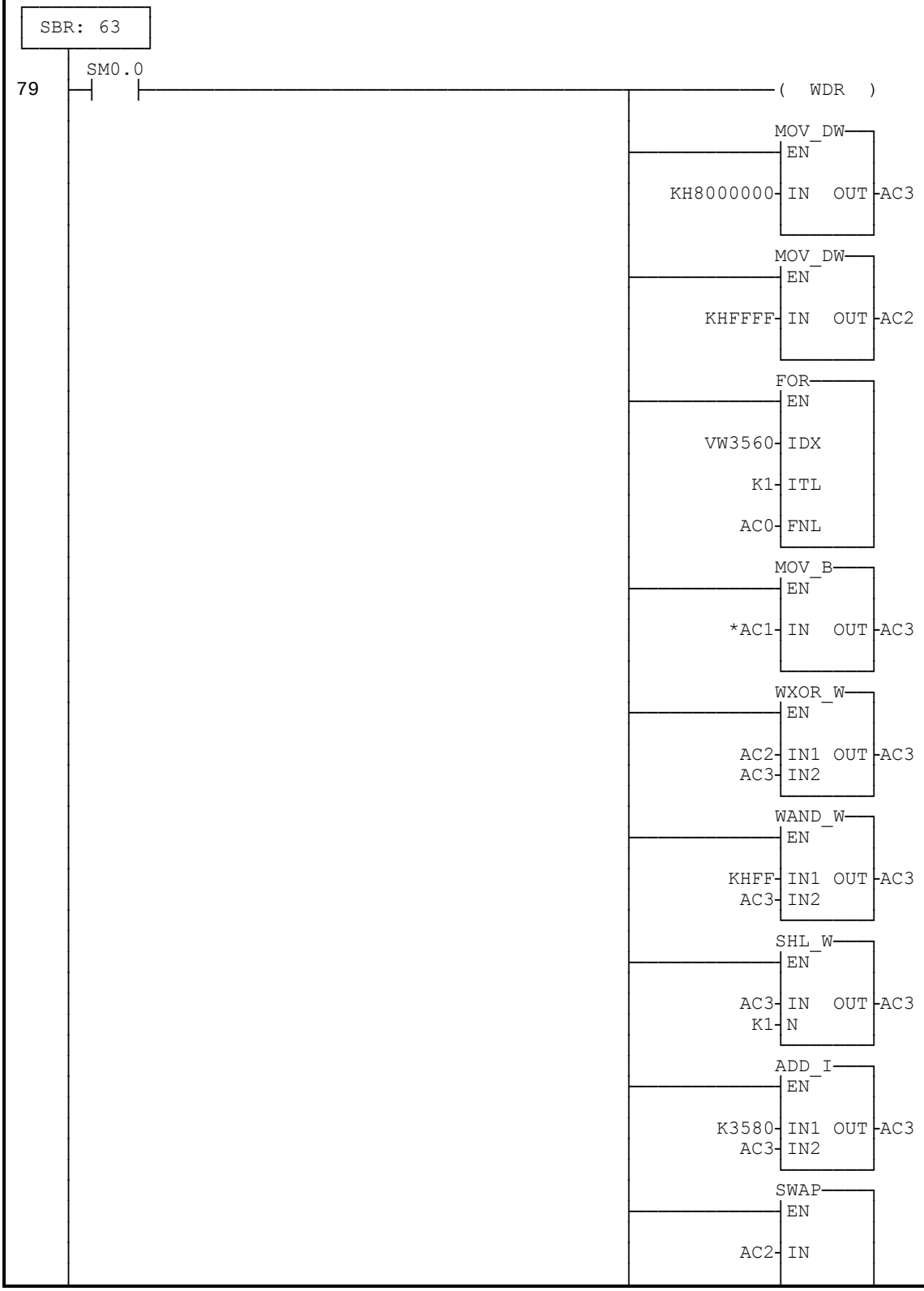
// Подпрограмма 62
//
// Эта подпрограмм генерирует таблицу CRC, используемую для подсчета значения CRC
// в телеграме.
//
// Используется многочлен: x16 + x15 + x2 + 1

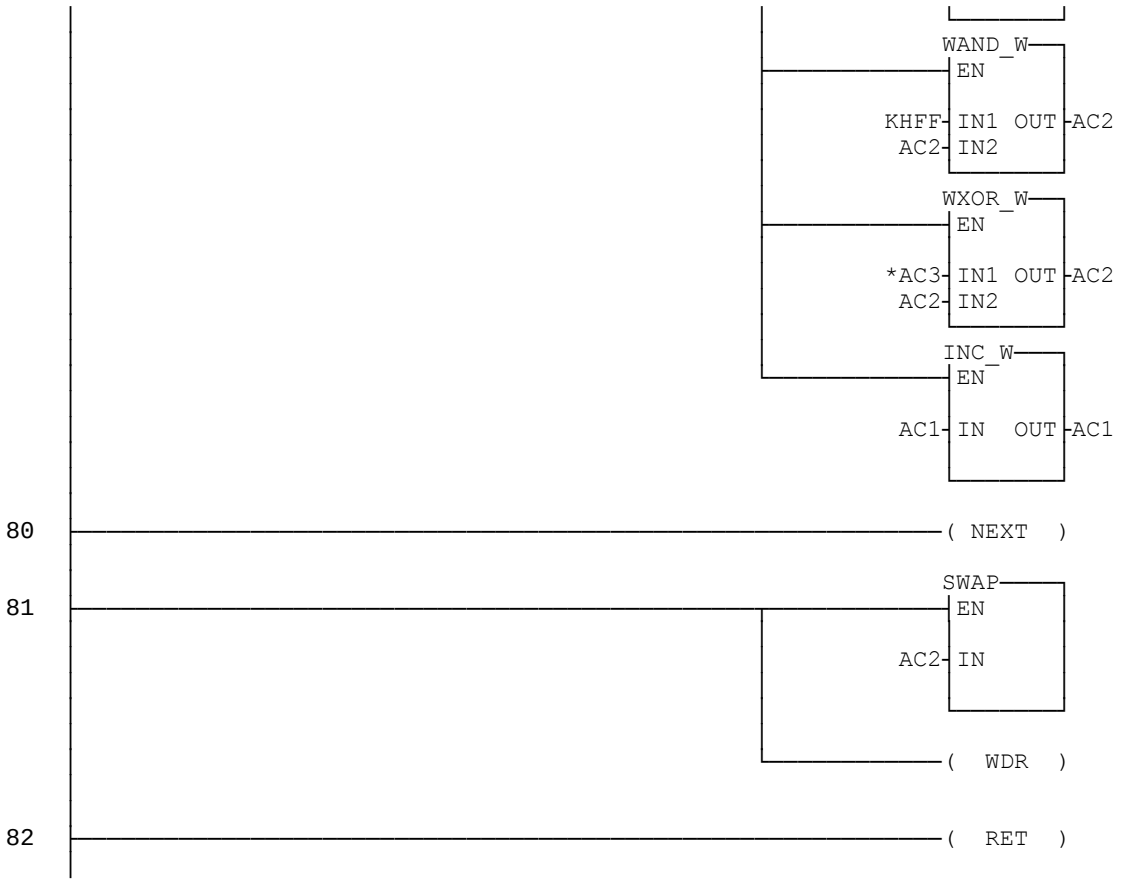





```
SBR         62
LD          SM0.0
MOVD        &VB3580,AC3          // загрузить указатель таблицы
MOVD        0,AC2                // загрузить индекс

FOR         VW3560,1,256         // для всех возможных значений байт
MOVW        AC2,AC0              //   значение = индекс
FOR         VW3562, 1, 8         //   для восьми бит из байта
SRW         AC0,1                //     сдвиг младшего бита
LD          SM1.1                //     если сдвинутый бит = 1
XORW        16#A001,AC0          //     XOR CRC со значением
NEXT                             //   конец для

LD          SM0.0
INCW        AC2                  //   увеличить индекс на единицу
MOVW        AC0,*AC3             //   запомнить слово таблицы
+I          2,AC3                //   увеличить на единицу указатель таблицы
WDR                              //   сбросить контрольный таймер
NEXT                             // конец для

RET
```

```
// Подпрограмма 63
//
// Данная подпрограмма расчитывает значение CRC для телеграммы, используя быстрый метод
// поиска в таблице.
//
// Входные парметры:    AC0     длина телеграммы
//                      AC1     указатель на телеграмму
//
// Выходные парметры:   AC2     значение CRC в младшем слове
```

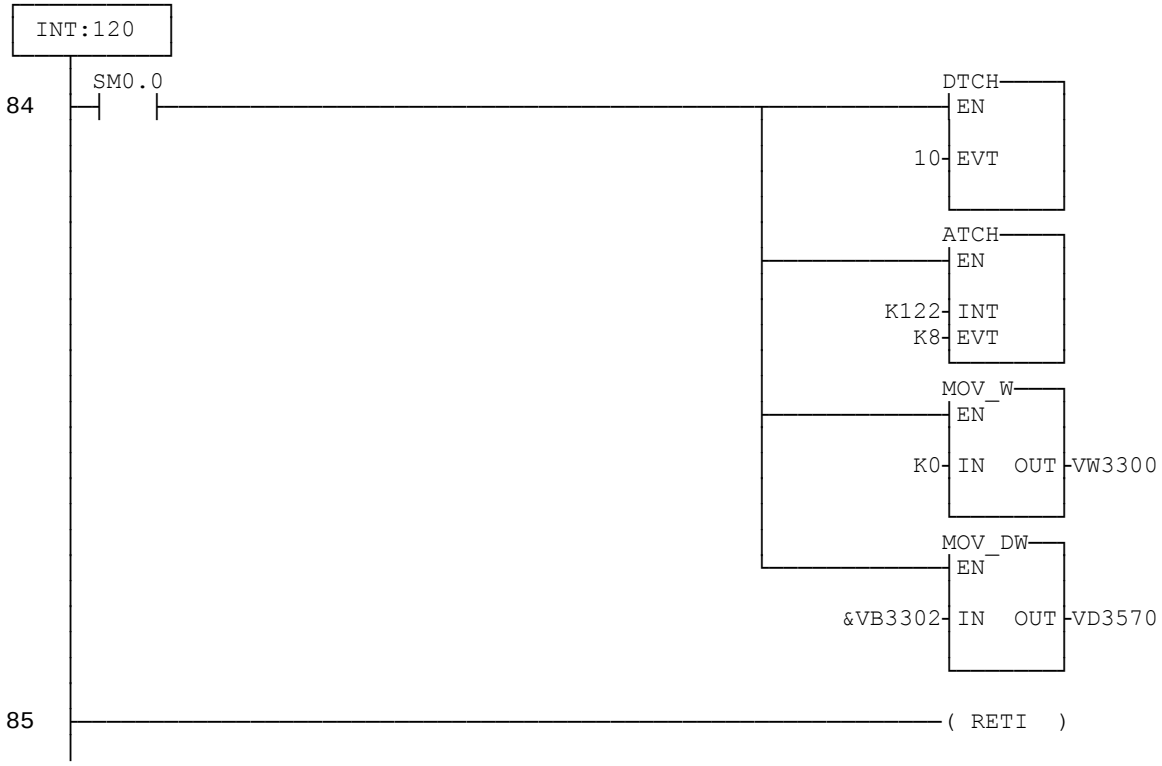





```
SBR     63
LD      SM0.0
WDR                         // сбросить контрольный таймер
MOVD    16#08000000,AC3     // очистить временный регистр
MOVD    16#0000FFFF,AC2     // инициализировать значение CRC = 0xFFFF

FOR     VW3560, 1, AC0      // для всех байт в телеграмме
MOVB    *AC1,AC3            //   получить байт телеграммы
XORW    AC2,AC3             //   XOR данные с CRC
ANDW    16#00FF,AC3         //   сохранить только младший байт
SLW     AC3,1               //   сконвертировать в индекс в слово таблицы
+I      3580,AC3            //   добавить начальный адрес таблицы (VB3580)
SWAP    AC2                 //   переставить байты в CRC
ANDW    16#00FF,AC2         //   сохранить только младший байт
XORW    *AC3,AC2            //   OR значение таблицы с CRC
INCW    AC1                 //   указатель на следующий байт телеграммы
NEXT                        // конец для

SWAP    AC2                 // переставить байты в CRC перед возвратом
WDR                         // сбросить контрольный таймер
RET
```

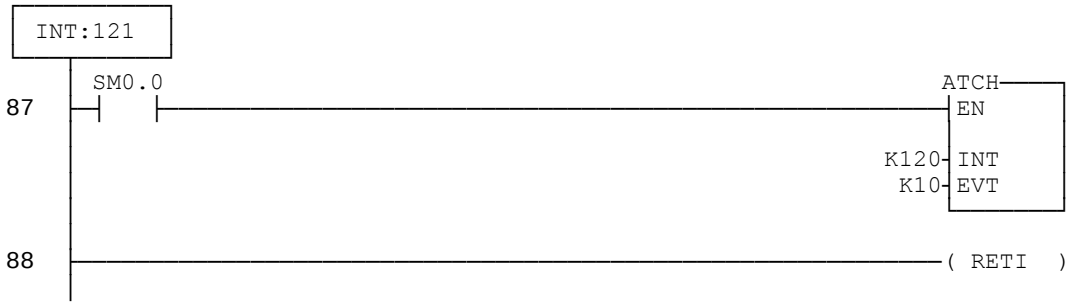
Программы обработки прерываний

```
// INT 120
//
// Данная программа обработки прерывания исполняется, когда превышен таймер
// свободной линии. Если это произошло, то присоединить программу прерываний для
// получения знака из порта, т.к. следующий принятый знак д.б. началом телеграммы.
```



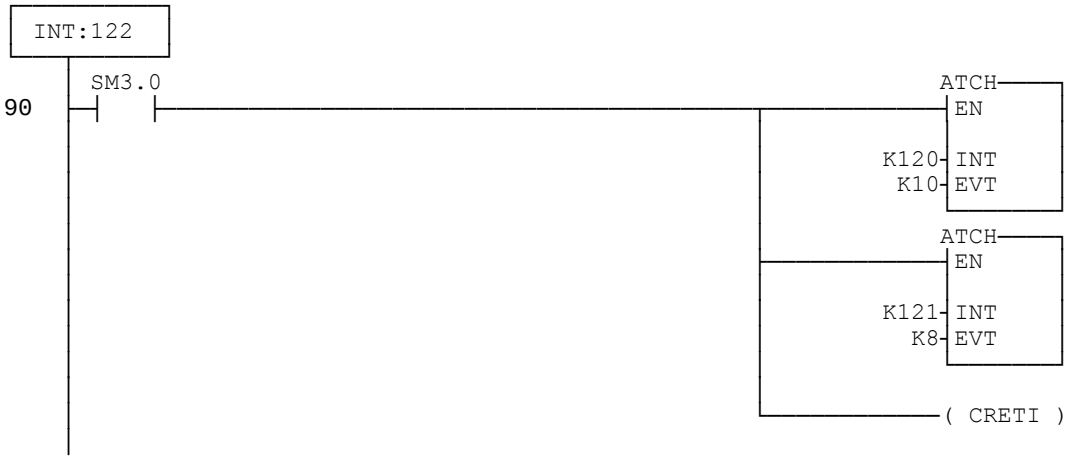

```
INT     120
LD      SM0.0
DTCH    10              // блокировать таймер свободной линии
ATCH    122,8           // назначить программу приема
                        // к коммуникационному порту
MOVW    0,VW3300        // установить счетчик приема в ноль
MOVD    &VB3302,VD3570  // установить указатель на буфер приема
RETI
```

```
// INT 121
//
// Данная программа обработки прерывания обрабатывает прерывания последовательного порта // для
обнаружения свободной линии. Если во время обнаружения свободной линии принят
// символ, то перезапустить таймер свободной линии.
```

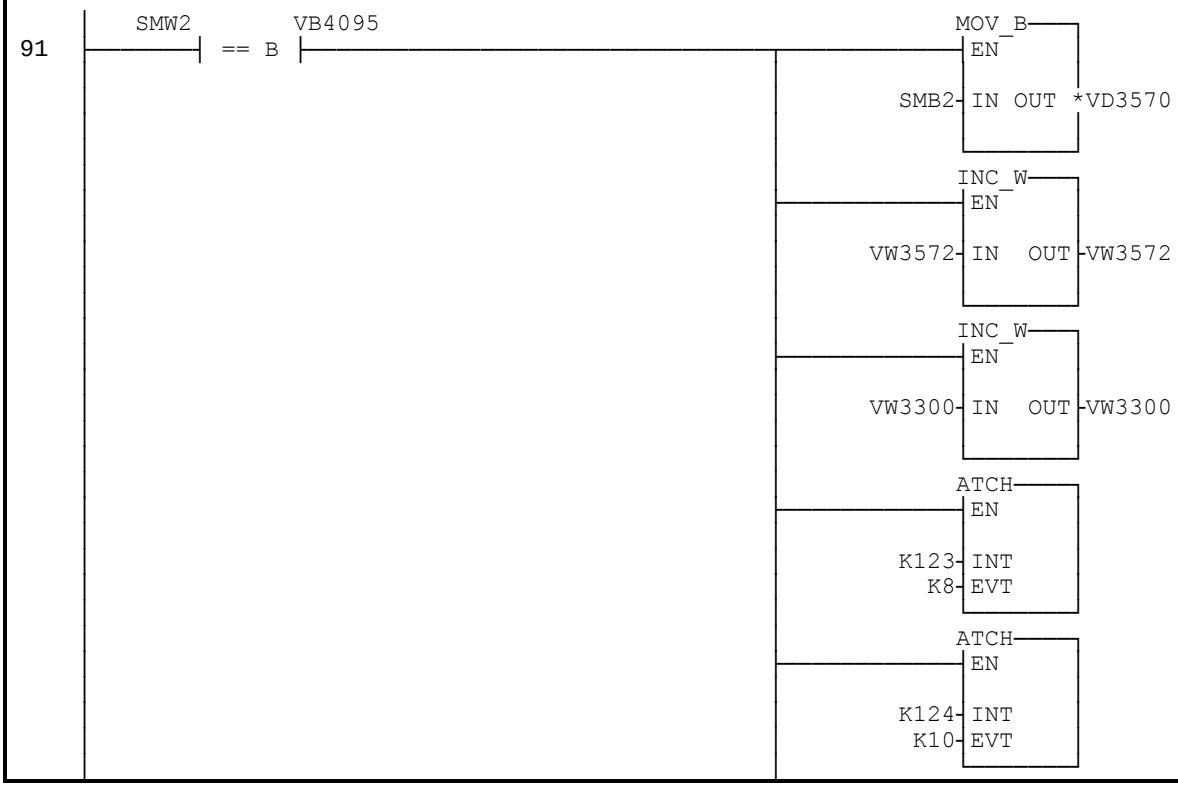



```
INT         121
LD          SM0.0
ATCH        120,10          // повторное разрешение таймера свободной линии
RETI
```

```
// INT 122
//
// Данная программа обработки прерывания принимает первый байт телеграммы после
// обнаружения времени свободной линии. Первый байт телеграммы - это адрес. Проверка,
// этот запрос для данного адреса. Если да, то присоединить INT 123 для приема всей
// телеграммы. Если телеграмма не для данного адреса, то возвратиться назад для
// поиска времени свободной линии.
//
// ЗАМЕЧАНИЕ:    широковещательный адрес в данной реализации не поддерживается,
//               т.к. он требует программирования дополнительного кода. Если данная
//               функция требуется, то д.б. модифицирована данная INT вместе со всеми
//               остальными функциями обработки, т.к. широковещательный адрес
//               не поддерживается всеми этими функциями.
```



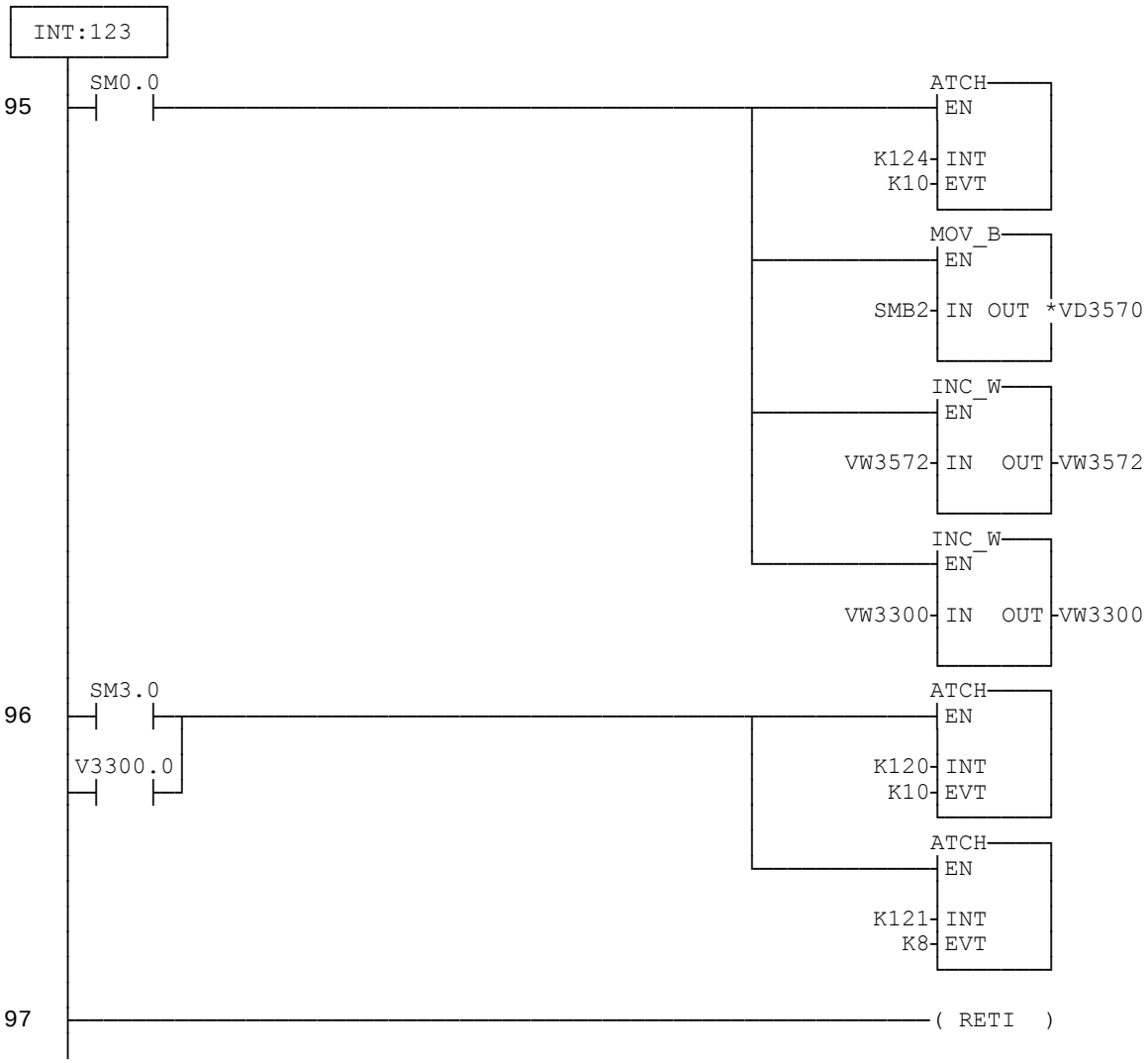

```
INT       122
LD        SM3.0          // если (ошибка четности)
ATCH      120,10         //   начальный поиск свободной линии
ATCH      121,8          //   INT 1, если мы получили символ
CRETI                    //   возврат
```




```
                                    // иначе (четность - OK)
LDB=    SMB2,VB4095                 //   если (адрес == мой адрес)
MOVB    SMB2,*VD3570                //     запомнить адрес в буфере
INCW    VW3572                      //     увеличить на 1 указатель буфера
INCW    VW3300                      //     увеличить на 1 счетчик принятых байтов
ATCH    123,8                       //     принять остаток телеграммы
ATCH    124,10                      //     конец приема, если тайм-аут
CRETI                               //     возврат
```

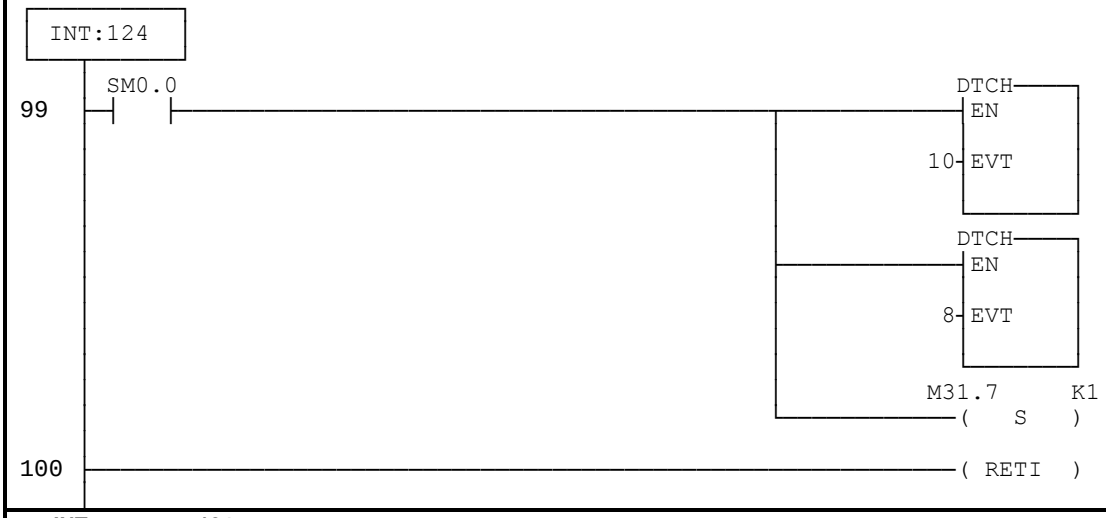

```
LD      SM0.0                       //   иначе (адрес != мой адрес)
ATCH    120,10                      //     повторное разрешение таймера свободной линии
ATCH    121,8                       //     INT 121, если мы получили символ
RETI
```

```
// INT 123
//
// Данная программа обработки прерывания обрабатывает тело телеграммы.
// Символы принимаются, контролируется ошибка четности и затем они помещаются в
// коммуникационный буфер. Количество байт увеличивается на 1 для каждого символа.
// Если число принятых байт больше, чем 256, то прием прерывается.
```



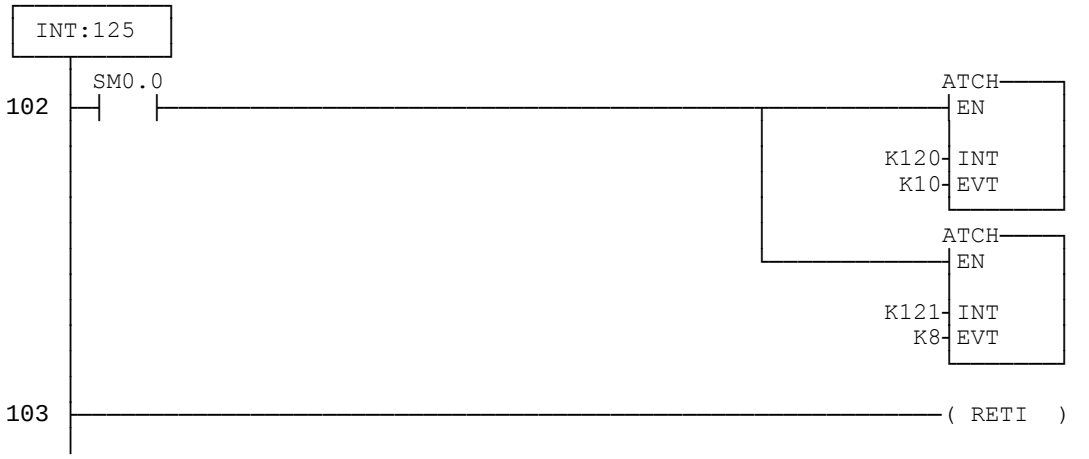

```
INT     123
LD      SM0.0
ATCH    124,10          // телеграмма закончена после следуещего тайм-аута MOVB
        SMB2,*VD3570    // запомнить байт телеграммы в буфере
INCW    VW3572          // увеличить на 1 указатель буфера
INCW    VW3300          // увеличить на 1 счетчик принятых байтов

LD      SM3.0           // если (ошибка четности) или
O       V3300.0         //   (переполнение буфера)
ATCH    120,10          //   начальный поиск свободной линии
ATCH    121,8           //   INT 1, если мы получили символ
RETI
```

```
// INT 124
//
// Данная программа обработки прерывания исполняется, когда была принята телеграмма
// и обнаружена еще раз свободная линия, это означает, что запрос завершен.
// Выполняемые действия - блокировать таймер свободной линии, блокировать прерывание
// порта и установить меркер фоновой обработки телеграммы.
```




```
INT     124
LD      SM0.0
DTCH    10      // блокировать таймер свободной линии
DTCH    8       // блокировать прерывание порта
S       M31.7,1 // установить меркер фоновой обработки
RETI
```

```
// INT 125
//
// Данная программа обработки прерывания исполняется, когда ответ был полностью
// передан обратно активной станции. Сброс системы прерывания для повторного поиска
// свободной линии и начала нового запроса.
```




```
INT     125
LD      SM0.0
ATCH    120,10  // начальный поиск свободная линия
ATCH    121,8   // INT 1 если получили символ
RETI
```

Указания по преобразованию

Для того чтобы преобразовать IEC STL в S7-Micro/DOS STL

- Добавьте ‘K’ перед каждым числом, не являющимся шестнадцатеричной константой (например, 4 ⇒ K4)
- Замените ‘16#’ на ‘KH’ для всех шестнадцатеричных констант (например, 16#FF ⇒ KHFF)
- Поставьте запятые для смены полей. Используйте клавиши перемещения или клавишу TAB для перехода от поля к полю.
- Для преобразования программы S7-Micro/DOS STL в LAD-форму каждый сегмент должен начинаться со слова ‘NETWORK’ и номера. Кажiый сегмент в этом примере имеет свой номер на диаграмме LAD. Используйте команду INSNW в меню редактора для ввода нового сегмента. Команды MEND, RET, RETI, LBL, SBR и INT требуют отдельных сегментов.
- Комментарии строк, обозначенные "//" не поддерживаются в S7-Micro/DOS, но разрешены комментарии сегментов

Общие указания

Примеры применения SIMATIC S7-200 предназначены для того, чтобы дать пользователям S7-200 начальную информацию, как можно решить с помощью данной системы управления определенные задачи. Данные примеры применения S7-200 бесплатны.

В приведенных примерах программ речь идет об идеях решения без претензии на полноту или работоспособность в будущих версиях программного обеспечения S7-200 или STEP7 Micro. Для соблючения соответствующих технически безопасных предписаний при применении необходимо предпринять дополнительные меры.

Ответственность Siemens, все равно по каким правовым нормам, при возникновении ущерба из-за применеия примеров программ исключается, равно и при ущербе личным вещам, персональном ущербе или при намеренных или грубо неосторожных действиях.