Функция 23 (17_{16}) Read/Write Multiple Registers

Даня функция производит операцию чтения и записи за одну MODBUS транзакцию. Функция может записывать новые значения Holding (выходных/внутренних) регистров и возвращать содержание другой группы Holding регистров. Эта функция поддерживается не всеми устройствами Modbus (уточняйте в документации к устройству).

В запросе указывается начальный адрес и количество регистров группы для чтения, после чего указывается начальный адрес, количество регистров, и данные для записи. Счетчик байтов перед данными для записи указывает на количество байт, передаваемых в поле данных.

Запрос:

Код функции	17_{16}
Адрес начального регистра группы чтения (Hi)	от 0 до $FFFF_{16}$
Адрес начального регистра группы чтения (Lo)	
Количество читаемых регистров (Hi)	от 1 до 76_{16} (118)
Количество читаемых регистров (Lo)	
Адрес начального регистра группы записи (Hi)	от 0 до $FFFF_{16}$
Адрес начального регистра группы записи (Lo)	
Количество регистров для записи (Hi)	от 1 до 76_{16} (118)
Количество регистров для записи (Lo)	
Счетчик байт	2*N
Данные для записи (1-й регистр Hi)	от 0 до $FFFF_{16}$
Данные для записи (1-й регистр Lo)	
…	…
Данные для записи (N-й регистр Hi)	от 0 до $FFFF_{16}$
Данные для записи (N-й регистр Lo)	

В ответе при отсутствии ошибок передаются считываемые регистры.

Ответ.

Код функции	17_{16}
счетчик байт	2*N
Считываемые данные (1-й регистр Hi)	от 0 до $FFFF_{16}$
Считываемые данные (1-й регистр Lo)	
…	…
Считываемые данные (N-й регистр Hi)	от 0 до $FFFF_{16}$
Считываемые данные (N-й регистр Lo)	

Пример 1. MODBUS. Запрос на чтение/запись значения Holding регистров.

Задача. Сформировать сообщение-запрос и сообщение-ответ на чтение Holding регистров с 108-го по 110-й (при условии адресации с 0), и одновременно записи регистров с 200-го по 201-й при положительной обработке запроса сервером.

	HI	LO	HI	LO	HI	LO	HI	LO		201-й		202-й	
17	00	6C	00	03	00	C8	00	02	04	CD	6B	00	05
функция	адрес начального регистра чтения		количество читаемых регистров		адрес начального регистра записи		количество регистров для записи		счетчик	значение регистров для записи			

сообщение-запрос

Рис.1. Формат сообщения-запроса

		108-й		109-й		110-й	
17	06	CD	6B	00	05	00	05
функция	счетчик	значение регистров для записи					

сообщение-ответ

Рис.2. Формат сообщения-ответа, если запрос обработан без ошибок

Пример 2. MODBUS. Создание функционального блока для реализации функции Read/Write Multiple Registers в Modicon M340.

Задача. Создать производный тип функционального блока (DFB Type) для реализации функции Modbus 17_{16} в M340, при условиях:

- количество читаемых и пишущих регистров произвольно в диапазоне от 1 до 20;
- управляющий вход START_REQ функции запускает запросы с максимально-возможной производительностью;
- данные для записи передаются в виде массива INT;
- прочитанные данные получаются в виде массива INT;
- предвидеть возможность контроля результата выполнения операции через выходы: номер коммуникационной ошибки (0 – отсутствие ошибки), номер ошибки Modbus протокола (0 – отсутствие ошибки), количество прочитанных байт;

Работу функционального блока продемонстрировать на примере периодического (период=500 мс) считывания с Slave 1, 6-ти регистров, начиная с 12741, и записи 2-х регистров, начиная с 12761.

Решение.

В UNITY PRO создаем DFB Type со структурой, показанной на рис.3. Кроме перечисленных в постановке задачи входов и выходов блока, дополнительно используется параметр типа input/output GEST_RW. Массив GEST_RW – соответствует массиву коммуникационных параметров, используемой в функциональном блоке коммуникационной функции DATA_EXCH.

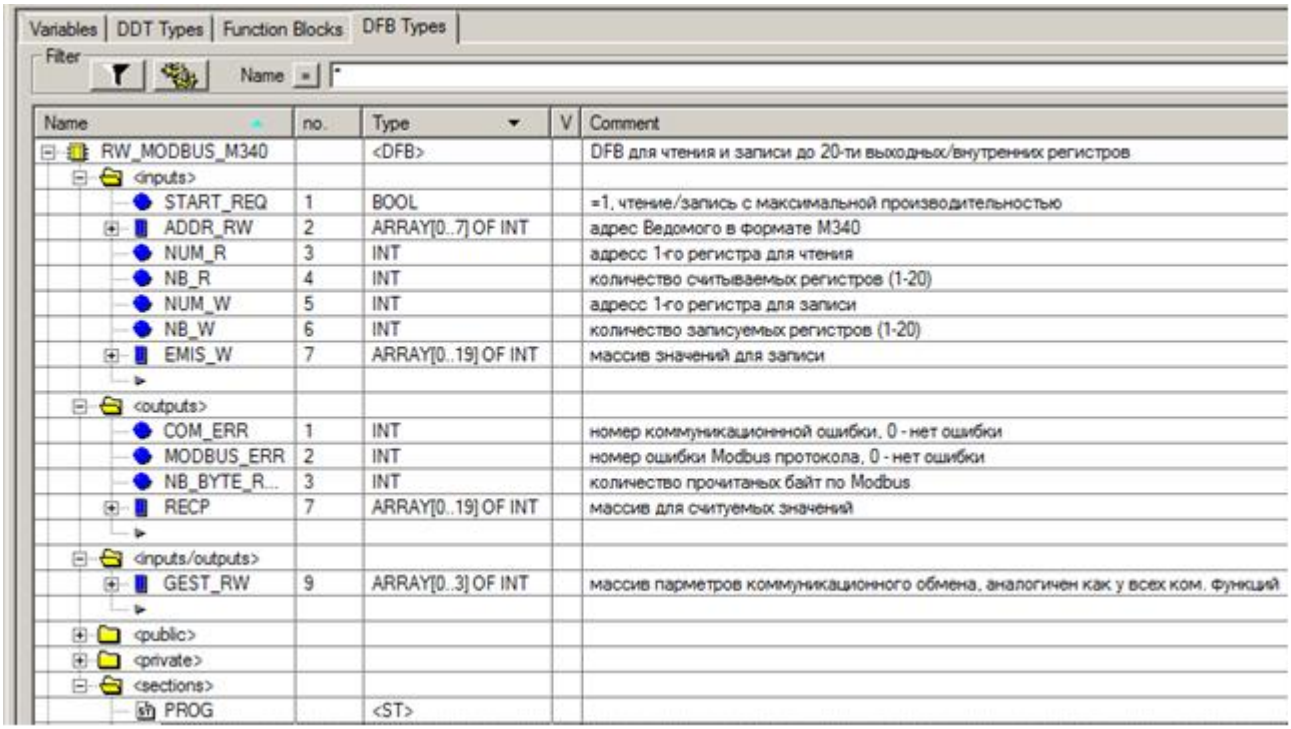

Рис.3. Структура DFB Type RW_MODBUS_M340.

Адрес Modbus Slave передается в блок том же формате, как и во всех коммуникационных функциях UNITY для M340.

Листинг секции PROG типа производного функционального блока RW_MODBUS_M340:

```
 if NOT GEST_RW[0].0 AND STARTED_REQ then (*если функция DATA_EXCH обработана*)
     STARTED_REQ:=FALSE;
     COM_ERR:=GEST_RW[1]; (*в 1-м слове ком параметров содержится отчет о обработке функции*)
     if COM_ERR=0 then (*если функция обработана без ошибок*)
             if (RECV_BUFER[0] and 16#00FF)=16#17 then (*если 1-й байт ответа равен номеру ф-ции*)
                NB_BYTE_READ:=ROL((RECV_BUFER[0] and 16#FF00),8); (*количество прочитаных байт во 2-м байте*)
                (*--------- значение считуемых регистров --------------*)
                for INDEX:=0 to NB_BYTE_READ/2 do
                     RECP[INDEX]:=ROL((RECV_BUFER[INDEX+1]),8); (*преобразование BIG/LITTLE ENDIAN *)
                end_for;
             else (*если 1-й байт не равен номеру функции, то вернулась ошибка*)
                MODBUS_ERR:=ROL((RECV_BUFER[0] and 16#FF00),8); (*номер ошибки во 2-м байте ответа*)
             end_if;
     end_if;
 end_if;

if START_REQ and NOT GEST_RW[0].0 then (*---------------- запуск запроса--------------*)
     SEND_BUFER[0]:= 16#0017 or NUM_R and 16#FF00;(*1байт-адр 1-го рег чтения (HI);2байт-функция*)
     SEND_BUFER[1]:= NUM_R and 16#00FF;  (*3байт-адр 2-го рег чтения (LO), 4байт-количество (HI) всегда = 0*)
     SEND_BUFER[2]:= NB_R and 16#00FF or (*5байт-количество рег чтения (LO)*)
                     NUM_W and 16#FF00;  (*6байт-адр 1-го рег для записи (HI)*)
     SEND_BUFER[3]:= NUM_W and 16#00FF;  (*7байт-адр 1-го рег для записи (LO), 8байт-кол (HI) всегда=0*)
     SEND_BUFER[4]:= NB_W and 16#00FF or (*9байт-количество рег для записи (LO)*)
                     NB_W*2*16#100;      (*10байт-кол последующих байт = кол_регистров*2 => в старш байт*)
     (*--------- значение регистров для записи --------------*)
     for INDEX:=0 to NB_W do
             SEND_BUFER[INDEX+5]:=ROL ((EMIS_W[INDEX]),8); (*преобразование BIG/LITTLE ENDIAN *)
     end_for;

     GEST_RW [3]:= NB_W*2+10;(*в последнее слово таблицы параметров нужно записать*)
                             (*количество передаваемых с запросом байт, за исключением адреса Slave*)
     (*----------- вызов функции ----------------*)
     DATA_EXCH (ADR := ADDR_RW ,
        TYP := 1, (*тип запроса, 1 - отправить запрос и ждать ответа*)
        EMIS :=SEND_BUFER  (*данные для передачи*),
        GEST := GEST_RW(*параметры управления*),
        RECP => RECV_BUFER (*данные для приема*));
     STARTED_REQ:=TRUE;(* метка запуска запроса*)
 end_if;
 if not START_REQ then  modbus_err:=0; end_if;
```

Для чтения регистров и битов можно воспользоваться функцией READ_VAR, для записи WRITE_VAR, для реализации остальных клиентских функций Modbus в M340 используется коммуникационная функция DATA_EXCH. Функция отсылает по адресу, указанному в ADR, запрос (содержание запроса записывается в SEND_BUFER). Параметр TYP=1 указывает, что функция должна ждать ответ. После позитивной обработки функции, ответ на запрос будет доступен в RECV_BUFER. Запрос формируется в формате MODBUS APP, то есть не включает поле адреса Slave (он берется с ADDR) и CRC контрольную сумму (она высчитывается автоматически). Для работы функции DATA_EXCH необходимо в последнее слово параметров управления коммуникационным обменом (GEST_RW[3]) записать количество предаваемых байт (исключая поле адреса и контрольной суммы).

Следует отметить, что в протоколе Modbus при передаче регистров сначала передается старший байт слова, а потом младший. Буферы для передачи DATA_EXCH передаются таким образом: младший байт предается первым. То-есть при передаче и приеме регистров, старшие и младшие байты будут поменяны местами (проблема совместимости BIG/LITLE Endian). Для того, чтоб решить эту проблему, в программе используется функция ROL, которая вращает влево указанное количество битов слов (8, т.е. один байт) в буфере передачи и приема. При заполнении буфера передачи, проходит смещение на один байт, по этому заполнение происходит побайтно.

Программа для периодического (период=500 мс) считывания с Slave 1, 6-ти регистров, начиная с 12741, и записи 2-х регистров, начиная с 12761, показана на рис.4.

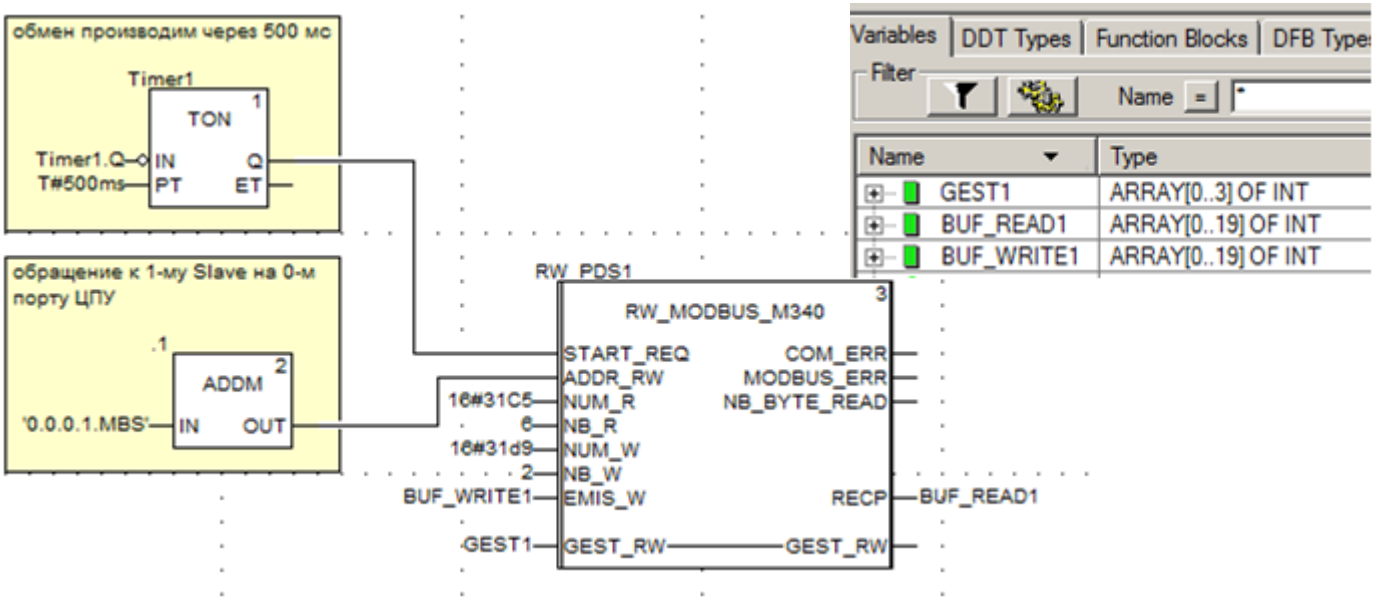

Рис.4. Программа для периодического считывания/записи регистров (слева) и таблица переменных (в правом верхнем углу)