DR. BOB DAVIDOV

Протокол ModBus-RTU / ASCII

Цель работы: освоение протоколов передачи данных промышленной сети.

Задача работы: построение связи компьютера с модулем дискретного ввода-вывода поддерживающего протокол ModBus.

Приборы и принадлежности: Персональный компьютер, модуль дискретного ввода-МК110, интерфейс RS-485, МатЛАБ, программа для передачи данных через COM порт, например HyperTerminal (start > All Programs > Acsessories > Communications >) или COM Port ToolKit.

ОБЩИЕ СВЕДЕНИЯ

Modbus – протокол последовательной передачи данных разработан компанией Modicon в 1979 году для программируемых логических контроллеров ПЛК. Устройство которое запрашивает информацию называется Modbus master. Устройство снабжающее информацией называется Modbus slaves. В сети modbus может находится один master и 247 slaves.

Версиями modbus протокола являются

- Modbus RTU
- Modbus ASCII
- Modbus TCP

Простейшее подключение modbus канала к последовательному порту показано на Рис. 1. Типовая скорость передачи 9600 бод (бит/сек). Максимальная скорость обмена по интерфейсу RS-485 составляет 115200 бит/сек.

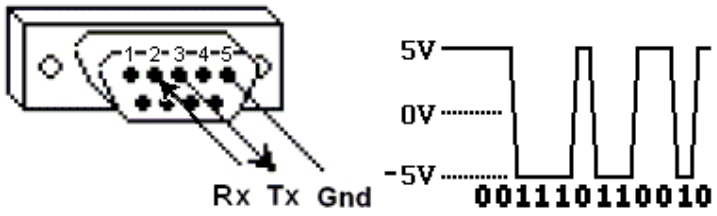


Рис. 1. Простейшее подключение modbus канала к последовательному порту.

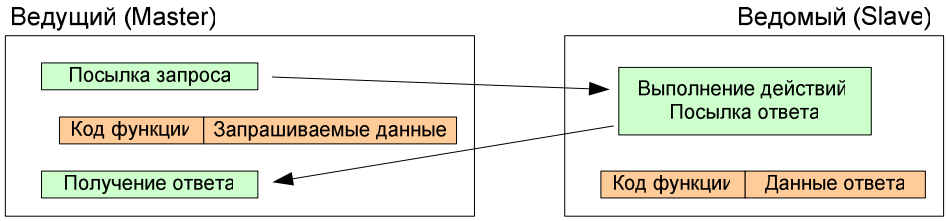


Рис. 2. Соединение Master – Slave.

В ответе передаются запрошенные данные. Количество байт данных зависит от количества запрошенных элементов. Перед данными передается один байт, значение которого равно количеству байт данных.

Значения регистров хранения и регистров ввода передаются начиная с указанного адреса, по два байта на регистр, старший байт каждого регистра передаётся первым:

байт 1	байт 2	байт 3	байт 4	…	байт N-1	байт N
RA,1	RA,0	RA+1,1	RA+1,0		RA+Q-1,1	RA+Q-1,0

			Function Name	Function Code
Data Access	Bit access	Physical Discrete Inputs	**Read Discrete Inputs**	2
		Internal Bits or Physical Coils	**Read Coils**	1
			Write Single Coil	5
			Write Multiple Coils	15
	16-bit access	Physical Input Registers	**Read Input Register**	4
		Internal Registers or Physical Output Registers	**Read Holding Registers**	3
			Write Single Register	6
			Write Multiple Registers	16
			Read/Write Multiple Registers	23
			Mask Write Register	22
			Read FIFO Queue	24
	File Record Access		Read File Record	20
			Write File Record	21
Diagnostics			Read Exception Status	7
			Diagnostic	8
			Get Com Event Counter	11
			Get Com Event Log	12
			Report Slave ID	17
			Read Device Identification	43
Other			Encapsulated Interface Transport	43

Информация в устройствах Modbus Slave хранится в четырех таблицах. Две таблицы содержат дискретные (On/Off) значения (coils) и две другие содержат численные значения (registers). И coils и registers содержат таблицы предназначенные только для чтения и для чтения / записи. Каждая таблица имеет размер 9999. Каждый Coil или Contact имеет один бит и адрес в диапазоне 0000 .. 270E. Размер каждого регистра – 16-бит слово.

Адрес первого Holding регистра – 0000

Адреса Modbus	**Размер данных**	**Тип**	**Описание**	**Код функции**
0 .. 9999	1 бит	Чтение - запись	Дискретные выходные Coils	1
10000..19999	1 бит	Только чтение	Дискретные входные контакты	2
30000..39999	16 бит	Только чтение	Аналоговые входные регистры	4
40000..49999	16 бит	Чтение-запись	Аналоговые выходные регистры временного хранения	3
60000..69999	16 бит		Дополнительные регистры (только для больших машин)	

MODBUS ASCII

Формат Modbus/ASCII

Старт	Адрес	Функция	Данные	Контроль ошибок	Конец
1 байт	2 байта	2 байта	n байт	2 байта	2 байта
Символ ‘:’	Адрес станции (устройства)	Код функции, например, считывание входных данных	адрес данных устройства затем – длина данных	Код LRC алгоритма равен отрицательной сумме предыдущих байт	Перевод строки (CR/LF)
3A	10	03			OD OA

Рассмотрим пример формирования записи команды запроса содержимого регистров аналоговых выходов (holding registers), адреса регистров: с 40108 по 40110. Адрес клиентского устройства: 17 ().

Рис. 3. Команда запроса. Контрольный байт равен сумме предыдущих байт (без стартового символа) взятый с отрицательным знаком или как “-Hex (255 - сумма) “.

```
3A 3131 3033 3030 3642 3030 3033 3745 0D 0A
```

Рис. 4. Команда запроса в ASCII шестнадцатеричных кодах (170 bits).

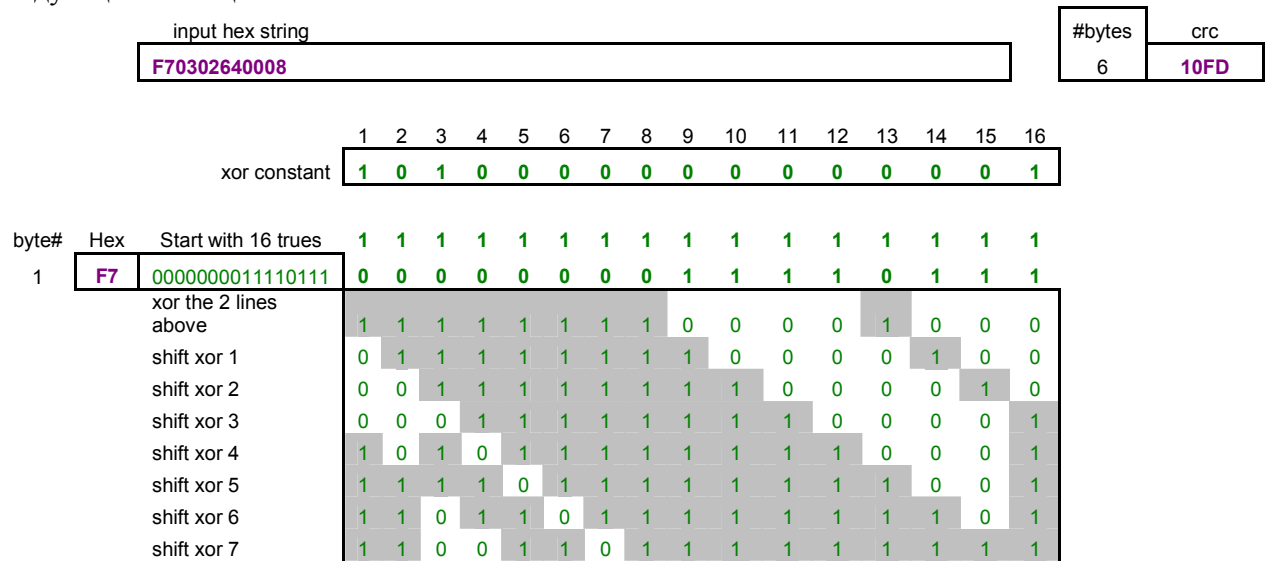
MODBUS RTU

Пример команды запроса (**Рис. 3**) в формате modbus RTU показан на **Рис. 5**. Код RTU почти в два раза меньше этой же команды запроса в формате ASCII.

```
11 03 00 6B 00 03 76 87
```

Рис. 5. Команда запроса (**Рис. 3**) в RTU шестнадцатеричных кодах. (80 bits). `76 87` – `контрольная сумма.`

Циклический алгоритм вычисления контрольной суммы RTU версии modbus протокола показан в следующей таблице.



		shift xor 8	1	1	0	0	0	1	1	0	1	1	1	1	1	1	1	0	FEC6
2	03	0000000000000011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
		xor the 2 lines above	1	1	0	0	0	1	1	0	1	1	1	1	1	1	0	1	
		shift xor 1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	
		shift xor 2	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	0	
		shift xor 3	0	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	
		shift xor 4	1	0	0	1	0	0	0	0	0	1	1	0	1	1	1	0	
		shift xor 5	0	1	0	0	1	0	0	0	0	0	1	1	0	1	1	1	
		shift xor 6	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	
		shift xor 7	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	
		shift xor 8	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0781
3	02	0000000000000010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
		xor the 2 lines above	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	
		shift xor 1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	
		shift xor 2	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	
		shift xor 3	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	
		shift xor 4	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	
		shift xor 5	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	
		shift xor 6	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	
		shift xor 7	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	
		shift xor 8	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	4103
4	64	0000000001100100	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	
		xor the 2 lines above	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0	1	
		shift xor 1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	1	1	
		shift xor 2	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0	
		shift xor 3	0	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	
		shift xor 4	0	0	1	1	1	1	0	0	0	0	1	1	0	0	1	0	
		shift xor 5	0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	1	
		shift xor 6	1	0	1	0	1	1	1	1	0	0	0	0	1	1	0	1	
		shift xor 7	1	1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	
		shift xor 8	1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	0	C2DB
5	00	0000000000000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		xor the 2 lines above	1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	0	
		shift xor 1	0	1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	
		shift xor 2	1	0	0	1	0	1	1	0	1	1	1	1	0	0	0	1	
		shift xor 3	1	1	1	0	1	0	1	1	0	1	1	1	1	0	0	1	
		shift xor 4	1	1	0	1	0	1	0	1	1	0	1	1	1	1	0	1	
		shift xor 5	1	1	0	0	1	0	1	0	1	1	0	1	1	1	1	1	
		shift xor 6	1	1	0	0	0	1	0	1	0	1	1	0	1	1	1	0	
		shift xor 7	0	1	1	0	0	0	1	0	1	0	1	1	0	1	1	1	
		shift xor 8	1	0	0	1	0	0	0	1	0	1	0	1	1	0	1	0	5A91
6	08	0000000000001000	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
		xor the 2 lines above	1	0	0	1	0	0	0	1	0	1	0	1	0	0	1	0	
		shift xor 1	0	1	0	0	1	0	0	0	1	0	1	0	1	0	0	1	
		shift xor 2	1	0	0	0	0	1	0	0	0	1	0	1	0	1	0	1	
		shift xor 3	1	1	1	0	0	0	1	0	0	0	1	0	1	0	1	1	
		shift xor 4	1	1	0	1	0	0	0	1	0	0	0	1	0	1	0	0	
		shift xor 5	0	1	1	0	1	0	0	0	1	0	0	0	1	0	1	0	
		shift xor 6	0	0	1	1	0	1	0	0	0	1	0	0	0	1	0	1	
		shift xor 7	1	0	1	1	1	0	1	0	0	0	1	0	0	0	1	1	
		shift xor 8	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	0	10FD

Программа МатЛАБ вычисления контрольной суммы Modbus RTU последовательности

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% crc_calculator.m     v1.0a
% Matlab v7.0 (R14) SP 1
% Bob Davidov
% 25 February 2012
%
% CRC algorithm
% calculates check sum of Modbus RTU sequence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output_hex_string = crc_calculator (Input_hex);
%Input_hex = 'F70302640008'; % <= 2 * 16 Char
F            = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
xor_constant = [1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1];
for i = 1 : length (Input_hex) / 2;
    A = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    if ~(i > length (Input_hex)/2)
        A_hex = Input_hex ((i-1)*2+1:i*2);  % Two HEX bytes
        A_bin = dec2bin (hex2dec (A_hex));
        length_A_bin = length (A_bin);
        for j = 0 : length_A_bin - 1
            A (16 - j) = str2num(A_bin (length_A_bin - j));
        end
    end
    F = xor (F,A);
    for ii = 1 : 8
        if F(16) ==1
            if xor_constant (1) == 0
                F_shift (1) = 0;
            else
                F_shift (1) = 1;
            end
            for j = 2 : 16;
                if xor_constant (j) == F (j-1);
                    F_shift (j) = 0;
                else
                    F_shift (j) = 1;
                end
            end
        else
            F_shift = circshift(F',1)';
        end
    F = F_shift;
    end
end
h = num2str(F);
h = h(1:3:length(h));
output_hex_string = num2str([dec2hex(bin2dec(h(9:12)))
dec2hex(bin2dec(h(13:16))) dec2hex(bin2dec(h(1:4)))
dec2hex(bin2dec(h(5:8)))]);
% End of crc_calculator.m
```

СРАВНЕНИЕ MODBUS/ASCII И MODBUS/RTU

Параметр	Modbus/ASCII		Modbus/RTU	
Используемые символы	ASCII 0..9 и A .. F		Двоичные 0 .. 255	
Проверка ошибки	LRC Longitudinal Redundancy Check		CRC Cyclic Redubdancy Check	
Начало фрейма	Символ ‘:’		Пауза 3.5 символа	
Конец фрейма	Символы CR/LF		Пауза 3.5 символа	
Интервал между сообщениями	1 сек.		Пауза 1.5 символа	
Стартовый бит	1		1	
Кол. бит данных	7		8	
Паритет	even/odd	none	even/odd	none
Стоповый бит	1	2	1	2

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ModBUS_RTU_ASCII_Convertor.m     v1.0a
% Matlab v7.0 (R14) SP 1
% Bob Davidov
% 23 February 2012
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

%Useful function:
%>>hex2dec('7531')
%>>char(36) => '%'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Address = 17; % Station Address 8 bit:  1..255
Function = 3; % see the table below
%
Data_First_Address = 107; % 16 bit, First address of output holding register
of 40108 is 107
Data_End_Address = 109; % 16 bit, First address of output holding register of
40108 is 107

Data_Chars = '';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Input data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if Data_First_Address < 10000
    Offset = 0;
    Function_check = 1;
elseif (Data_First_Address >= 10000) && (Data_First_Address < 20000)
    Offset = 10000;
    Function_check = 2;
elseif (Data_First_Address >= 30000) && (Data_First_Address < 40000)
    Offset = 30000;
    Function_check = 4;
elseif  (Data_First_Address >= 40000) && (Data_First_Address < 50000)
    Offset = 40000;
    Function_check = 3;
```

```
end

%Data_First_SubAddress = Data_First_Address - Offset - 1;
Data_length = Data_End_Address - Data_First_Address + 1;          % 16 bit
Lenght of holding registers from 40108 to 40110 is 3
Data_First_SubAddress = Data_First_Address - Offset;

% Modbus Addresses      Data size   Description        Function Code
% 0..9999               1 bit       Output (coils)         1
% 10000..19999          1 bit       Inputs (relays)        2
% 30000..39999          16bit       Analog inputs          4
% 40000..49999          16bit       Holding registers      3
% 60000..69999          16bit       Extended Registers (added to spec for
larger machines only)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculation of ASCII request
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Error_checks = 256-(Address + Function + Data_First_SubAddress +
Data_length);

A1 = dec2hex(double(':'));

if Address < 16
    A2 = dec2hex(double(['0' num2str(dec2hex(Address))]));
else
    A2 = dec2hex(double(num2str(dec2hex(Address))));
end

if Function < 16
    A3 = dec2hex(double(['0' num2str(dec2hex(Function))]));
else
    A3 = dec2hex(double(num2str(dec2hex(Function))));
end

dfa = num2str(dec2hex(Data_First_SubAddress));
l = length(dfa);
str = '0000';
str(4-l+1:4) = dfa;
A4 = dec2hex(double(str));

dfa = num2str(dec2hex(Data_length));
l = length(dfa);
str = '0000';
str(4-l+1:4) = dfa;
A5 = dec2hex(double(str));

Data_Chars_req = [];
for i = 1 : length (Data_Chars) / 2;
    Data_Chars_req = [Data_Chars_req dec2hex(double(Data_Chars((i-1)*2+1)))
dec2hex(double(Data_Chars(i*2))) ' '];
end

A6 = dec2hex(double(num2str(dec2hex(Error_checks))));

%The complete ASCII HEX request is made by first adding the message
delimiting characters.
disp(sprintf('Station N%2d; Function: %2d); Data address: %4d .. %4d (Offset
: 4d)',Address,Function,Offset,Data_First_Address,Data_End_Address));
disp(sprintf('             Data length:%4d, Data_Chars:
%s',Data_length,Data_Chars));
ASCII_request =  [A1 ' ' A2(1,1) A2(1,2) A2(2,1) A2(2,2) ' ' A3(1,1) A3(1,2)
```

```
A3(2,1) A3(2,2) ...
                    ' ' A4(1,1) A4(1,2) A4(2,1) A4(2,2) ' ' A4(3,1) A4(3,2)
A4(4,1) A4(4,2) ...
                    ' ' A5(1,1) A5(1,2) A5(2,1) A5(2,2) ' ' A5(3,1) A5(3,2)
A5(4,1) A5(4,2) ...
                    ' ' A6(1,1) A6(1,2) A6(2,1) A6(2,2) ' ' Data_Chars_req
'0D' ' ' '0A'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculation of RTU request
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%The equivalent Modbus RTU message would be:
if Address < 16
    A2 = ['0' num2str(dec2hex(Address))];
else
    A2 = num2str(dec2hex(Address));
end

if Function < 16
    A3 = ['0' num2str(dec2hex(Function))];
else
    A3 = num2str(dec2hex(Function));
end

dfa = num2str(dec2hex(Data_First_SubAddress));
l = length(dfa);
str = '0000';
str(1,4-l+1:4) = dfa;
A4 = str;

dfa = num2str(dec2hex(Data_length));
l = length(dfa);
str = '0000';
str(1,4-l+1:4) = dfa;
A5 = str;

A6 = crc_calculator ([A2 A3 A4 A5 Data_Chars]);
%A6 = crc_calculator ('F70302640008')

Data_Chars_req = [];
for i = 1 : length (Data_Chars) / 2;
    Data_Chars_req = [Data_Chars_req Data_Chars((i-1)*2+1:i*2) ' '];
end

RTU_request =  [A2(1,1) A2(1,2) ' ' A3(1,1) A3(1,2) ...
            ' ' A4(1,1) A4(1,2) ' ' A4(1,3) A4(1,4) ...
            ' ' A5(1,1) A5(1,2) ' ' A5(1,3) A5(1,4) ...
            ' ' Data_Chars_req ...
            A6(1,1) A6(1,2) ' ' A6(1,3) A6(1,4)]

% End of ModBUS_RTU_ASCII_Convertor.m
```

MODBUS RTU КОМАНДЫ СВЯЗИ С МОДУЛЕМ ВВОДА ВЫВОДА **ОВЕН МК110**

Адрес модуля (устройства) – 16 (заводская установка). Полный список регистров ModBus и примеры командной последовательности для модуля МК110 приведены в следующей таблице.

Табл. 1. Регистры протокола ModBus.

Параметр	Ед. измерен	Значение	Тип	Адрес регистра (hex)	Адрес регистра (dec)	Примеры команды для модуля МК110
Значение на выход №1	0.1 %	0… 1000	uint16	0000	0000	10 10 00 00 00 01 02 00 00 66 00 запись в рг.0 ШИМ 0 10 10 00 00 00 01 02 01 F4 66 17 запись в рг.0 ШИМ 500 10 03 00 00 00 01 87 4B чтение регистра ШИМ
Значение на выход №2	0.1 %	0… 1000	uint16	0001	0001	10 10 00 01 00 01 02 00 00 67 D1 запись в рг.1 ШИМ 0 10 10 00 01 00 01 02 01 F4 67 C6 запись в рг.1 ШИМ 500 10 03 00 01 00 01 D6 8B чтение регистра ШИМ
Значение на выход №3	0.1 %	0… 1000	uint16	0002	0002	10 10 00 02 00 01 02 00 00 67 E2 запись в рг.2 ШИМ 0 10 10 00 02 00 01 02 01 F4 67 F5 запись в рг.2 ШИМ 500 10 03 00 02 00 01 26 8B чтение регистра ШИМ
Значение на выход №4	0.1 %	0… 1000	uint16	0003	0003	10 10 00 03 00 01 02 00 00 66 33 запись в рг.3 ШИМ 0 10 10 00 03 00 01 F2 88 ответ: запись выполнена 10 10 00 03 00 01 02 01 F4 66 24 запись в рг.3 ШИМ 500 10 10 00 03 00 01 F2 88 ответ: запись выполнена 10 03 00 03 00 01 77 4B чтение регистра ШИМ 10 03 02 00 00 44 47 ответ: 2 байт рег. = 0

						10 03 02 01 F4 44 50 ответ: 2 байт рег. = 500
Аварийное значение на выходе №1	0.1 %	0… 1000	uint16	0010	0016	
Аварийное значение на выходе №2	0.1 %	0… 1000	uint16	0011	0017	
Аварийное значение на выходе №3	0.1 %	0… 1000	uint16	0012	0018	
Аварийное значение на выходе №4	0.1 %	0… 1000	uint16	0013	0019	10 03 00 13 00 01 76 8E чтение рег. 19
						Задается в 0..100% и определяет скважность ШИМ
Период ШИМ на выходе №1	сек	1… 900	uint16	0020	0032	
Период ШИМ на выходе №2	сек	1… 900	uint16	0021	0033	
Период ШИМ на выходе №3	сек	1… 900	uint16	0022	0034	
Период ШИМ на выходе №4	сек	1… 900	uint16	0023	0035	10 03 00 23 00 01 76 81 чтение рег. 35
Макс. сетевой тайм-аут	сек	0… 600	uint16	0030	0048	
Битовая маска значений выходов	–	0… 15	uint16	0032	0050	10 03 00 32 00 01 26 84 чтение битовой маски адр 50
Битовая маска значений входов	–	0… 255	uint16	0033	0051	10 03 00 33 00 01 77 44 чтение битовой маски адр 51
Значение счетчика входа №1	срабатывание	0… 65535	uint16	0040	0064	10 03 00 40 00 01 86 9F чтение счетчика адр 64
Значение счетчика входа №2	срабатывание	0… 65535	uint16	0041	0065	10 03 00 41 00 01 D7 5F чтение счетчика адр 65
...						
Значение счетчика входа №8	срабатывание	0… 65535	uint16	0047	0071	10 03 00 47 00 01 37 5E чтение счетчика адр 71
Логика управления выходом №1	–	0… 7	uint16	0050	0080	

Логика управления выходом – №2	0… 7	uint16	0051	0081	
Логика управления выходом – №3	0… 7	uint16	0052	0082	
Логика управления выходом – №4	0… 7	uint16	0053	0083	10 03 00 53 00 01 77 5A чтение рег. 83 0 – управляется только по интерфейсу RS-485 (ШИМ) 1 – (прямая логика) значение на выходе равно значению на входе (ШИМ не выполняется) 2 – (функция “НЕ”) значение на выходе равно инверсному значению на входе 3 – (функция “И”) задается для 2 входов и двух выходов 4 – (функция “ИЛИ”) задается для 2 входов и 2 выходов 5 – (один импульс) при включении входа (по переднему фронту) на выходе импульс заданной длительности 6 – (ШИМ импульс) при включенном входе на выход выдается ШИМ 7 – (Триггер) задается для 2 входов и двух выходов If вход1=1 и вход2=0 то выход1=выход2=1 If вход2=1 то выход1=выход2=0
Тип задержки управления - выходом №1	0… 2	uint16	0060	0096	
Тип задержки управления - выходом №2	0… 2	uint16	0061	0097	
Тип задержки управления -	0… 2	uint16	0062	0098	

выходом №3						
Тип задержки управления выходом №4	-	0… 2	uint16	0063	0099	10 03 00 63 00 01 77 55 чтение рег. 63 0 - нет задержек 1 – установлена задержка включения выхода 2 - установлена задержка выключения выхода
Задержка управления выходом №1/длина импульса на выходе №1	x 0,1 [сек]	0… 65535	uint16	0070	0112	
Задержка управления выходом №1/длина импульса на выходе №2	x 0,1 [сек]	0… 65535	uint16	0071	0113	
Задержка управления выходом №1/длина импульса на выходе №3	x 0,1 [сек]	0… 65535	uint16	0072	0114	
Задержка управления выходом №1/длина импульса на выходе №4	x 0,1 [сек]	0… 65535	uint16	0073	0115	10 03 00 73 00 01 76 90 чтение рег. 73 0 – не используется 1, 2, 3, 4 – задается время задержки в 0.1 долях сек. 5,6 – задается длина импульса в 0.1 долях сек. 7 – не используется

Примечание.

1) Запись в регистры осуществляется командой 16 (0x10), чтение – командами 03 или 04 (прибор поддерживает обе команды).

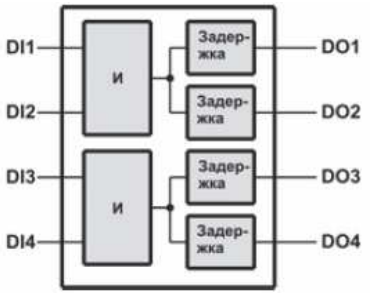
2) Обнуление счетчиков делается записью 0 в регистры хранения результатов счета.

3) В регистрах битовых масок значений входов и выходов старший бит соответствует входу или выходу с наибольшим номером: (бит, равный 1, соответствует состоянию выхода «Включено» и входа «Замкнут»).

4) Период ШИМ задается через конфигуратор. Заводская установка периода – 1сек. Минимальная длительность ШИМ – 50 мс.

ГРУППОВЫЕ КОМАНДЫ МК110

10 03 00 40 00 08 46 99 – чтение содержимого всех счетчиков модуля МК110.
10 03 00 00 00 04 47 48 – чтение содержимого всех регистров ШИМ.



- пример структуры функции “И”

Полезные МатЛАБ команды передачи / приема данных через COM порт.

Команда	**Назначение**
s = serial ('COM1');	создание COM объекта порта последовательной передачи данных
fopen(s); fclose(s)	подключение (отсоединение) объекта к серверу
get(s)	считывание и отображение параметров COM
s.BaudRate	объекта
set (s, 'BaudRate', 9600)	установка параметров
s.BaudRate = 2400	
fprintf(s,'RS232?')	запись данных
fscanf(s)	чтение данных
delete(s);	удаление объекта из Workspace памяти

Конфигурационные параметры модуля МК110 (заводские установки):

- Интерфейс: RS-485
- Скорость обмена данными: 9600 бод,
- Длина слова данных: 8 бит
- Тип контроля четности данных: no,
- Количество стоп-бит: 1 бит,
- Задержка ответа по сети: 2 мс,
- Базовый адрес модуля: 16;
- Период ШИМ: 1 сек.,
- Допустимые значения ШИМ: 1..900,

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Передача данных МатЛАБ в модуль МК110 по протоколу modbus RTU.

1. Подключите модуль МК110 к USB порту компьютера через преобразователь U110.

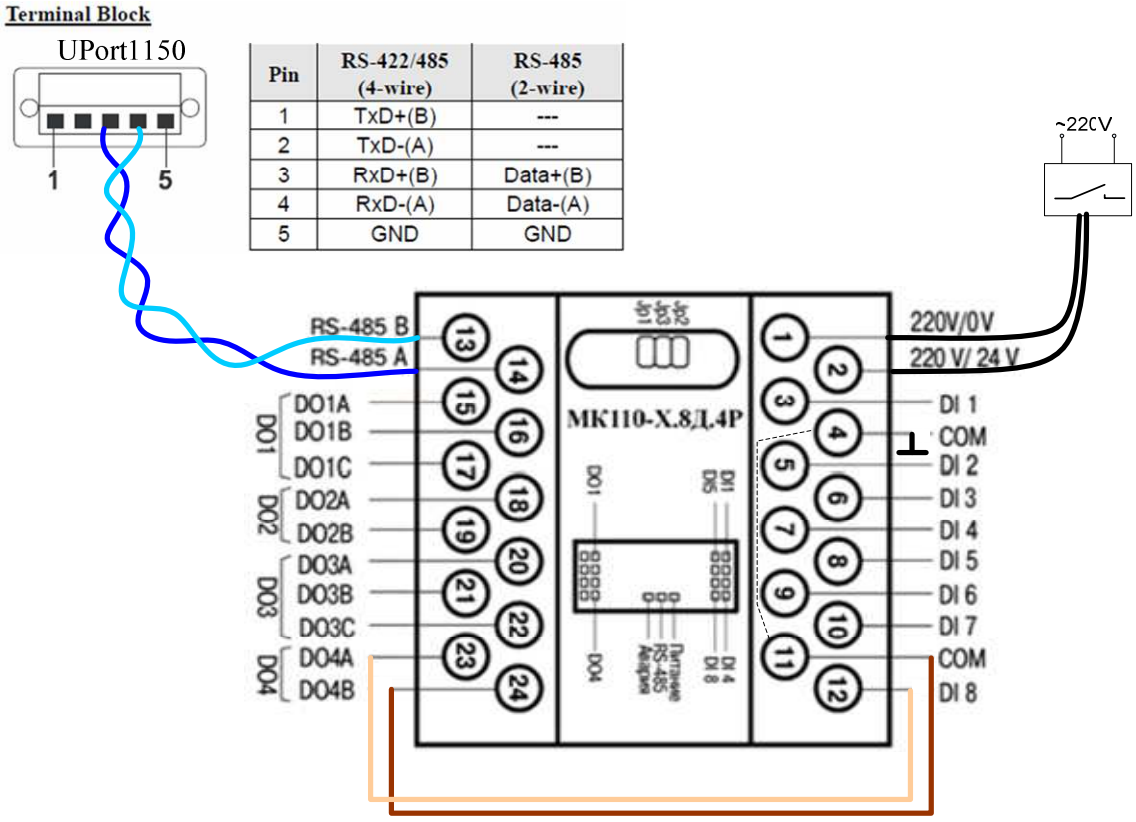


Рис. 6. Устройство дискретного ввода-вывода МК110-224.8Д.4Р. Схема подключения для выполнения работы.

2. Подключите RS-485 устройство к порту COM1.

   Start > My Computer > Properties > Hardware > Device Manager > Multi-port serial adapter > UPort1150 > Port Configuration > COM No

   Примечание: Если COM1 занят, отключите устройство от COM1 переводом параметра “Communications Port (COM1) Properties > Device usage” в состояние “Do not use this device (disable)”.

3. Настройте преобразователь U110 на интерфейс RS-485 2W

   Start > My Computer > Properties > Hardware > Device Manager > Multi-port serial adapter > UPort1150 > Port Configuration > Interface >

4. Загрузите МатЛАБ.

5. Сформируйте кодовую последовательность записи 500 в 4-ый регистр PWM модуля МК110, передайте последовательность в модуль и примите от него ответный код, например, как показано далее.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MatLAB_RTU_commandar.m     v1.0a
% Matlab v7.0 (R14) SP 1
% Bob Davidov
% 28 February 2012
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

%Useful function:
%>>hex2dec('7531')
%>>char(36) => '%'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Com_Port_Num = 'COM1';

Address = 16; % Station Address 8 bit:  1..255
Function = 16; % 3 or 4 is read; 16 is write;
PWM = 00; % 0 .. 1000
Data_First_Address = 3; % First address of Module Rg
Address_Range = 1; %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Input data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dfa = dec2hex(Data_First_Address);
l = length(dfa);
str = '0000';
str(4-l+1:4) = dfa;
Data_First_Address_Bytes = [hex2dec(str(1:2)) hex2dec(str(3:4))];

dfa = dec2hex(Address_Range);
l = length(dfa);
str = '0000';
str(4-l+1:4) = dfa;
Address_Range_Bytes = [hex2dec(str(1:2)) hex2dec(str(3:4))];

if Function == 16
   dfa = dec2hex (PWM);
   l = length(dfa);
   str = '0000';
   str(4-l+1:4) = dfa;
   PWM_Bytes = [2 hex2dec(str(1:2)) hex2dec(str(3:4))];
else
   PWM_Bytes = [];
end

% Master's Tx data without Check sum
Code = [Address Function Data_First_Address_Bytes Address_Range_Bytes PWM_Bytes];

Code_Char = dec2hex(Code);
Code_Char_line = [];
for I = 1:length(Code)
   Code_Char_line = [Code_Char_line Code_Char(I,1:2)];
end

% Check sum calculation
Check_Sum = crc_calculator(Code_Char_line)

% Master's Tx data with Check sum
```

```
RTU_Code = [Code hex2dec(Check_Sum(1:2)) hex2dec(Check_Sum(3:4))]

COM1 = serial (Com_Port_Num);
fopen (COM1);
fwrite(COM1, RTU_Code);

pause(0.002);
BytesAvailable = get(COM1,'BytesAvailable');
if BytesAvailable > 0
   Rx = fread(COM1,BytesAvailable)'
end

fclose (COM1);
delete (COM1);

% End of MatLAB_RTU_commandar.m.m
```

6. Остановите PWM записью ноля в соответствующий регистр модуля МК110.
7. Считайте содержимое 8-го счетчика (адр 71) модуля, следующей командной последовательностью которую необходимо подать на модуль через COM порт.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные отличия протоколов modbus RTU и ASCII.
2. Назовите возможные области применения протокола modbus в среде МатЛАБ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. http://en.wikipedia.org/wiki/Modbus
2. http://www.simplymodbus.ca/
3. Официальная спецификация modbus протокола. www.modbus-ida.org