**Преобразователь Ethernet-RS485\_RS232**

Поддерживаемые протоколы:

DHCP – запросы к DHCP серверу и получение адреса

SNMPv1 – обработка запросов Get, Set, GetNext

ModbusTCP – работа в режиме слейва

ModbusRTU – работа в режиме мастера

UDP – посылка дейтаграмм на определенный адрес

1. **Выбор динамического или статического IP адреса:**

Для того чтобы устройство выполняло запрос IP адреса необходимо в файле Ethernet.h установить значение DHCP\_Enable в 1.

*uint8\_t* DHCP\_Enable=1;

Если необходим статический IP адрес, то нужно установить DHCP\_Enable=0; прописать значения адреса, маски подсети, основного шлюза в файле Ethernet.h

*uint8\_t* DHCP\_Enable=0;

*uint8\_t* static\_IP[4]={10,0,0,76};

*uint8\_t* static\_dhcpGatewayIp[4]={10,0,0,254};

*uint8\_t* static\_dhcpSubnetMask[4]={255,255,255,0};

1. **Управление устройством по SNMP:**

Для того чтобы получить доступ к переменным по SNMP протоколу необходимо заполнить масив:

SNMP\_FIELD SNMP\_VARS[] = {

{"1.3.6.1.2.1.1.1.0",SNMP\_SYNTAX\_OCTETS, locDescr, Text, READ},

{"1.3.6.1.2.1.1.2.0",SNMP\_SYNTAX\_OCTETS, locObjectID, Text, READ},

{"1.3.6.1.2.1.1.3.0",SNMP\_SYNTAX\_TIME\_TICKS,&locUpTime,Long\_Word, READ},

{"1.3.6.1.4.1.2017.12.1.0", SNMP\_SYNTAX\_INT32, &MB\_HoldReg[0],Word, READWRITE},

};

Где SNMP\_FIELD, - структура

typedef struct {

const char\* oid;

SNMP\_SYNTAXES syn;

void \* variable;

*uint8\_t* variableType;

*uint8\_t* AccessLevel;

}SNMP\_FIELD;

OID – строка идентификатор

Syn – синтаксис SNMP

\*variable – указатель на переменную

variableType – тип переменной в программе, сейчас доступны следующие типы

word – двухбайтное целое

Text – строка

Long\_Word – 32разрядное целое

Для корректной работы необходимы следующие соответствия:

|  |  |
| --- | --- |
| syn | variableType |
| SNMP\_SYNTAX\_OCTETS | Text |
| SNMP\_SYNTAX\_INT32 | Word |
| SNMP\_SYNTAX\_TIME\_TICKS | Long\_Word |

Если необходимы другие комбинации типов, то преобразования для них нужно описать в функциях **decode()** и **encode()**

**AccessLevel**

READ – только для чтения

READWRITE – чтение и запись

**3.Modbus TCP**

Устройство одновременно является мастером в сети

RS-485/ModbusRTU и слейвом в сети MODBUS TCP.

Мастер ModbusRTU работает так же как мастер на Вега 2.3.

Порт на котором будет работать ModbusTCP Slave задается в Modbus\_TCP\_Automat.h

#define TCP\_PORT 502

Карта переменных мастера в MODBUS\_RTU является одновременно картой переменных слейва в MODBUS\_TCP;

Все значения карты переменных

#define Modbus\_Map\_List { \

{ \

MB\_Coil, MB\_TCP\_COILS\_QT, \

MB\_Input, MB\_TCP\_DISCRETE\_INPUTS\_QT, \

MB\_HoldReg, MB\_TCP\_HOLDING\_REGISTERS\_QT, \

MB\_InReg, MB\_TCP\_INPUT\_REGISTERS\_QT \

}, \

}

Могут читатся по MODBUS TCP. Значения в котушках и регистрах хранения могут устанавливатся

**4.Посылка данных по UDP**

Устройство может слать данные посылками по UDP (По аналогии с текущими данными в контроллере фрикулинга), при этом, благодаря наличию четырех аппаратных сокетов это практически не влияет на остальные сервисы

Для отправки данных нужно в файле TEST\_AUTOMAT.h заполнить масив GPRS\_DATA