ЗАО «Сервотехника»

ВЗАИМОДЕЙСТВИЕ КОНТРОЛЛЕРА ВЕРХНЕГО УРОВНЯ С ПРИВОДАМИ СЕРИЙ СПС И СПШ

Версия документа 1.1

Москва 2015 год

Оглавление

1.1. Введение

Существует множество вариантов взаимодействия с приводами:

- USB порт (см. Рис. 1). Непосредственное взаимодействие с контроллером привода через микросхему Silicon Labs CP21xx виртуального COM порта по интерфейсу USB. Данный вариант является наиболее простым.
  Микросхема Silicon Labs CP21xx реализует интерфйсу USB 2.0. Производитель поставляет драйвера для Win32, Win64, WiinCE, MAC OS, Linux, Android.
  http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx.
  Данный вариант взаимодействия имеет ряд ограничений, связанный с низкой помехозащищенностью канала связи, дальностью связи (максимально 5 метров) и сложностью подключения большого количества приводов.

- CAN шина (см. Рис. 1). Непосредственное взаимодействие с контроллером привода через CAN шину. Данный вариант требует наличие CAN интерфейса в самом контроллере верхнего уровня. Например, для ПК потребуется дополнительный USB-CAN конвертор или карта расширения CAN-PCI. CAN шина является надежной и удобной шиной. CAN шина является высокозащищенным каналом связи. При ее использовании можно организовывать взаимодействие с большим количеством приводов (до 128) и увеличить дальность связи до 50метров на скорости 500 кБод.

- Ethernet шина (см. Рис. 1). В сервоприводы серии СПС встроен дополнительный контроллер связи, который представляет собой шлюз между шиной Ethernet и контроллером привода.
  Шина Ethernet является высоконадежной и широкополосным каналом связи. Шина позволяет подключать практически неограниченное количество приводов, удаленных практически на любое расстояние.
- Ethernet шина. Посредством использования Ethernet CAN шлюза ECG (см. Рис. 1). Дополнительный вариант для сервоприводов серии СПШ.
- Подключение набора приводов СПС/СПШ через один USB разъем.
  Данный режим позволяет подключить приводы, объединенные CAN шиной, по одному USB порту. При этом в приводе, к USB порту которого подключается ПК, устанавливается режим ретрансляции “ip16: USB2CAN конвертор”. При этом запускается протокол USB2CAN, который позволяет «видеть» все приводы, подключенные по CAN.
  Данный режим с некоторыми ограничениями позволяет относительно легко взаимодействовать с сетью приводов.
- Начиная с 2014 года сервоприводы СПС опционально комплектуются интерфейсом EtherCAT (см. Рис. 1).

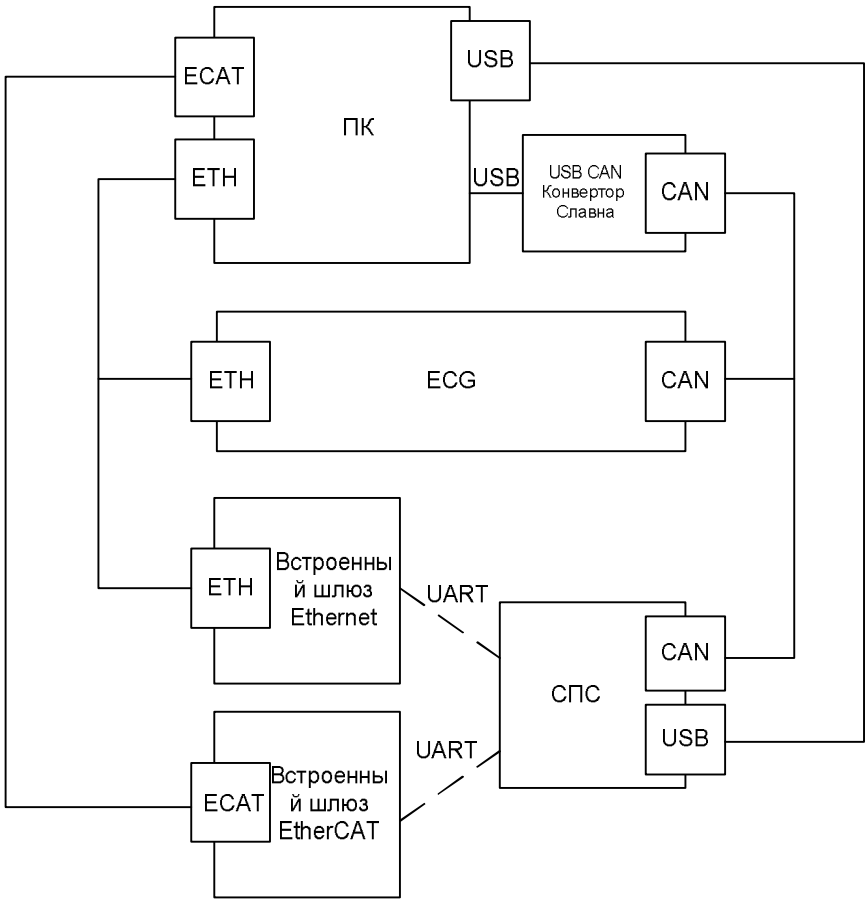

Рис. 1. Варианты интерфейсов контроллера верхнего уровня с приводами серии СПС.

Вследствие низкой стабильности работы USB в условиях производства, настоятельно не рекомендуется его использовать для оперативного управления. Данный протокол предназначен исключительно для настройки и диагностики приводов.

В комплекте поставки привода поставляется специализированная библиотека связи moto_dll2.dll. Данная библиотека позволяет реализовать любую из приведенных схем связи. Ограничением является режим подключения по CAN. В текущей версии moto_dll2.dll поддерживается только одна плата расширения CAN – преобразователь USB/CAN компании Славна (http://www.slavna.ru/stran/ucc06.htm). Библиотека автоматически обнаруживает все приводы, подключенные по описанным интерфейсам.

В ряде случаев использование библиотеки moto_dll2.dll невозможно. Например, используется ОС отличная от Windows. В данном случае необходимо самостоятельно реализовать протокол связи по определенному интерфейсу.

1.2. Взаимодействие с приводом по интерфейсу USB. Технологический протокол

Данный протокол носит название технологический протокол. Протокол предназначен для предоставления доступа к параметрам привода. Используя технологический протокол пользователь может запрашивать и устанавливать значения параметров.

Формат сообщения технологического протокола:

Табл. 1. Формат сообщения.

1 байт	2 байт	1-4 байт
Команда	Адрес параметра	Значение параметра

Допустимые команды технологического протокола:

Табл. 2. Список допустимых команд.

Команда	Краткое описание	Примечание
5	Запрос значения параметра	Команда передается внешним контроллером при запросе значения параметра
6	Передача текущего значения параметра	Команда передается приводом в ответ на команду 5
7	Установка значения параметра	Команда передается внешним контроллером при установке значения параметра
17	Запустить программу ПЛК	В поле «Адрес параметра» передается 0x0000. В Поле «Значение параметра» = номер банка программ в 16ти битном формате, из которого требуется запустить программу. Например, запустить программу из банка 1: Для останова текущей программы необходимо передать сообщение с «Значение параметра» =0x00FF;
22	Ретрансляция сообщения в CAN	Подробнее см. п. 1.6
23	Ретрансляция сообщения из CAN	Подробнее см. п. 1.6

Механизм прозрачности

Пакеты по технологическому протоколу передаются с обрамлением. Каждое сообщение обрамляется служебными символами Начало: 0x01 и Конец: 0x03. При этом в самом теле сообщения данные байты исключаются. Исключение байт происходит за счет их модификации, путём замены служебных символов на пару символов «Маркер замены», «Исходный символ +0x40».

Обрамление сообщение служебными символами позволяет обеспечить целостность канала связи, которая может нарушиться вследствие появления сбоя.

Порядок передачи

В поле «Адрес параметра» первым передаётся старший байт. В поле «Данные» первым передаётся младший байт.

Ниже приведены функции обрамления сообщения служебными символами и очистки от них.

Лист. 1. Листинг обрамления сообщения служебными символами и очисти от них.

```
#define RxBEGIN         0
#define RxDATA          1
#define RxEND           2

#define SOH             0x01
#define ETX             0x03
#define SUB             0x1a
```

```
/*****************************************************************************
Функция пересылки с предварительным обрамлением пакета служебными символами
*****************************************************************************
/
void TranspareSend(unsigned char input_byte)
{
   unsigned int t;

   if((input_byte == SOH)||(input_byte == ETX)||(input_byte == SUB)) {
          SCI_send(SUB);
          SCI_send(input_byte+0x40);
   }
   else {
          SCI_send(input_byte);
   }
}

/*****************************************************************************
Функция очистки принятого пакета от служебных символов
*****************************************************************************/
int ReTranspare(unsigned char *output_none_or_one_bytes, unsigned char input_byte)
{
    unsigned char ret_code = NONE_BYTE_RECEIVED;
    switch(ReTranspareStruct.Type)    {
    case RxBEGIN:
   if(input_byte == SOH) {
          ReTranspareStruct.Type = RxDATA;
          ReTranspareStruct.Ctrl = 0;
   }
      break;
    case RxDATA:
      switch(input_byte)      {
       case ETX: //Пакет получен
          ReTranspareStruct.Type = RxBEGIN;
                ret_code = PACKET_RECEIVED;
            break;
       case SOH:
            ReTranspareStruct.Type = RxBEGIN;
            break;
       case SUB:
             ReTranspareStruct.Ctrl = 1;
             break;
       default:
            if(ReTranspareStruct.Ctrl == 1)
            {
                      *output_none_or_one_bytes = input_byte - 0x40;
                ReTranspareStruct.Ctrl = 0;
            }
            else    *output_none_or_one_bytes = input_byte ;
                ret_code = ONE_BYTE_RECEIVED;
       }
            break;
     }
     return ret_code;
}
```

Пример отработки алгоритма.

Исходное сообщение		0x05	0x00	0x01		
Сообщение после преобразования	0x01	0x05	0x00	0x1A	0x41	0x03

Формат всех параметров приведен в документе «***Описание параметров***» и в файле «***tree_ru.dev***», который можно найти в папке установки программы МотоМастер.

Пример запроса уникального идентификатора устройства st3. Адрес параметра 0x001E.

0x01 0x05 0x00 0x1e 0x00 0x00 0x00 0x00 0x03

В ответ привод перешлет сообщение вида:

0x01 0x06 0x00 0x1e 0x00 0x02 0x04 0x05 0x03

Пример установки задания скорости ct2. параметр имеет размерность 4 байта. Формат IQ12 - дробный с фиксированной точкой, младшие 12 бит представляют дробную часть.

Устанавливаемое значение 550.25 об/мин. В формате IQ12:

550.25 * 4096 = 2253824 или в 16тиричном формате 0x00226400.

В итоге сообщение выглядит следующим образом:

0x01 0x07 0x02 0x02 0x00 0x64 0x22 0x00 0x03

Запуск программы ПЛК в банке 1:

0x01 0x14 0x00 0x00 0x1A 0x41 0x00 0x03

Останов ПЛК:

0x01 0x14 0x00 0x00 0xFF 0x00 0x03

1.3. Взаимодействие с приводом по интерфейсу CAN

Подробно протокол обмена по шине CAN описан в документе «***Описание параметров***» в разделе «***Протокол управления сервоприводами по шине CAN***».

В большинстве случаев наиболее правильной схемой является использование того же технологического протокола, который описан в предыдущем разделе. Данный протокол обрамляется в CAN сообщение. При этом обрамление служебными символами и механизм прозрачности не используется, т. к. протокол CAN самостоятельно обеспечивает целостность сообщения на канальном уровне.

Сообщение CAN представляет собой фиксированную структуру данных, в которую входит идентификатор (ID) и 8 байт данных.

Идентификатор может быть стандартный 11 бит или расширенный 29 бит.

Табл. 3. Формат сообщения CAN.

ID	D0	D1	D2	D3	D4	D5	D6	D7
Идентификатор 11 или 29 бит	Байт 0	Байт 1	Байт 2	Байт 3	Байт 4	Байт 5	Байт 6	Байт 7

Идентификатор сообщения формируется разбит на 3 логических переменных.

Табл. 4. Структура стандартного идентификатора CAN 11 бит.

Биты идентификатора ID	**10..6**	**5 .. 3**	**2..0**
Переменная	Команда (com)	Адрес источника (source_addr)	Адрес назначения (sink_addr)

Адрес источника – это адрес отправителя сообщения.

Адрес приемника – адрес устройства, которому предназначено данное сообщение.

В каждой изолированной сети все устройства должны иметь уникальный адрес.

В рамках одной сети необходимо, чтобы каждое устройство имело уникальный адрес в диапазоне от 0 до 127.

Приводы с адресами 0-6 воспринимают только пакеты со стандартным идентификатором.

Адрес 7 используется как широковещательный. Любая посылка, переданная со значением поля source_addr, равным 7, будет воспринята всеми приводами.

Для взаимодействия с приводами, которые имеют адрес в диапазоне 8-127, используется расширенный идентификатор.

Табл. 5. Структура расширенного идентификатора CAN 29 бит.

Биты идентификатора ID	**29..14**	**13 .. 7**	**6..0**
Переменная	Команда (com)	Адрес источника (source_addr)	Адрес назначения (sink_addr)

Пример кода на языке С для 32-битного компилятора, формирующий сообщение с командой 8 для адреса в диапазоне 0-6.

```
struct CAN_MSG {
        unsigned int id;
        unsigned char flags;
        unsigned char len;
        unsigned char data[8];
};
```

Код формирования:

```
*((int*)can_msg.data) = REQUIRED_POS;
*((int*)&(can_msg.data[4])) = FORCE_VELOCITY;
*((int*)&(can_msg.data[4])) <<= 5;
*((int*)&(can_msg.data[4])) &= 0xFFE0;
*((int*)&(can_msg.data[4])) |= (Enable<<2) | (P_OUT1) )<<1 | P_OUT0;
can_msg.id=0x200|((i+1)&7);//где i – адрес привода
can_msg.len=8;
can_msg.flags=0;//standard frame format
```

Примеры использования технологического протокола

Все ниже приведенные примеры для упрощения приведены до применения механизма прозрачности.

Запрос значения статуса привода с CAN адресом 1, параметра dd11

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x341	0x05	0x04	0x0F					

Так как адрес удаленного узла менее 7, то используется стандартный идентификатор 11 бит.

В ответ на данный запрос привод передаст в шину CAN команду – текущий статус привода.

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x388	0x06	0x04	0x0F	1	0			

В данном примере статус привода «Останов по команде».

Запрос значения статуса привода с CAN адресом 8, параметра dd11

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x34008	0x05	0x04	0x0F					

Так как адрес удаленного узла более 7, то используется расширенный идентификатор 29 бит.

В ответ на данный запрос привод передаст в шину CAN команду – текущий статус привода.

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x38400	0x06	0x04	0x0F	1	0			

В данном примере статус привода «Останов по команде».

Установка задания по скорости привода с CAN адресом 1, параметра ct2

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x341	0x07	0x02	0x02	0x00	0x60	0x09	0x00	

Значение передается в формате IQ12. Формат IQ12 определяет дробное число с фиксированной точкой. В формате IQ12 старшие 20 бит определяют целое знаковое число, а младшие 12 бит – дробную часть числа.

В данном примере устанавливаемое значение v=150 об/мин.

В формате IQ12 v_iq12 = v<<12=0x00096000.

Запуск программы ПЛК привода с CAN адресом 1, расположенной в банке программ 0, и останов

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x341	0x14	-	-	0x00	0x00			

ID	D0	D1	D2	D3	D4	D5	D6	D7
0x341	0x14	-	-	0xFF	0x00			

1.4. Взаимодействие с приводом по интерфейсу Ethernet через встроенный контроллер связи (шлюз)

При обмене с сервоприводами серии СПС по Ethernet интерфейсу используется «***Механизм прозрачности***» описанный выше. Механизм позволяет выделять сообщения в потоке передаваемых данных.

Чтобы определить все доступные устройства в сети, осуществляется их поиск с помощью широковещательного сообщения. Программа на ПК посылает сообщение на порт UDP номер 50024 (Рис. 2).

Все подключенные сервоприводы отвечают на данное сообщение, направляя ответ в порт 50025.

Прикладная программа ПК выполняет подключение к сокету TCP порта номер 50023, открытого в приложении в приводе.

Далее обмен между прикладной программой ПК и устройством осуществляется в соответствие с протоколом ETH2CAN, описанный ниже.

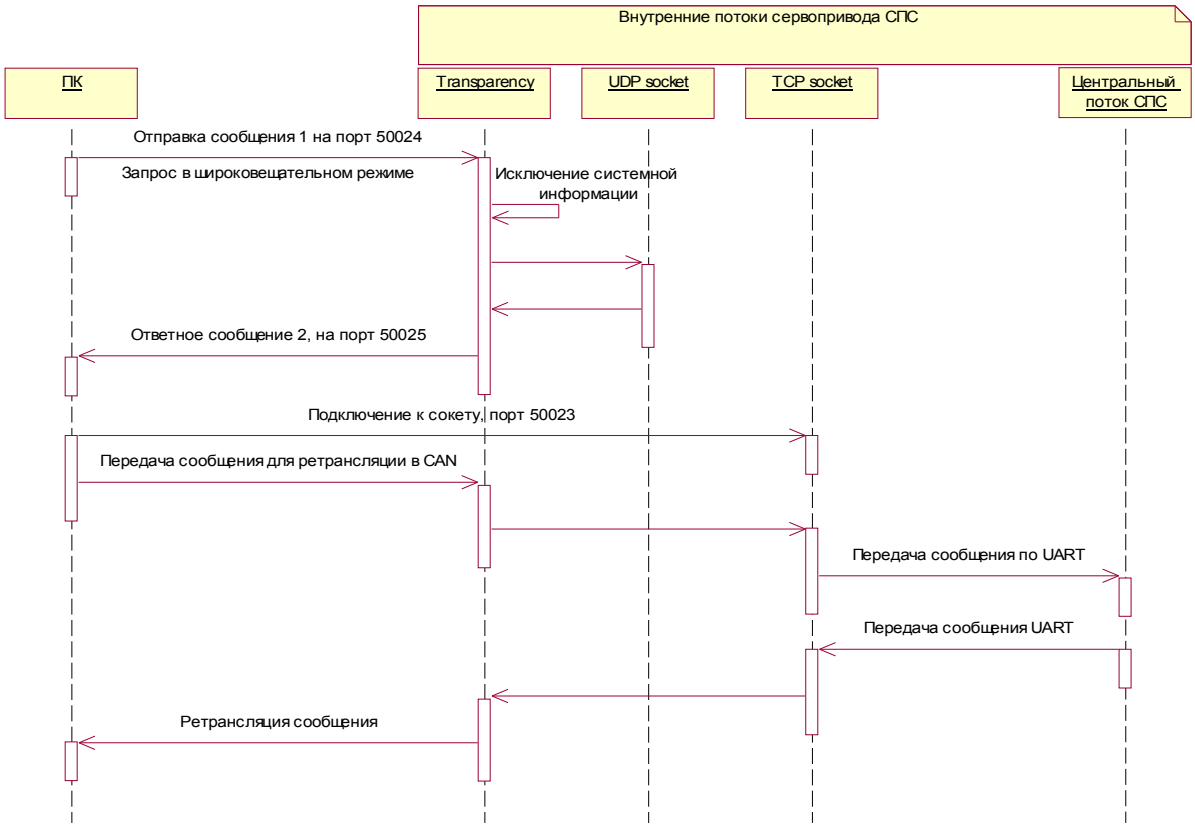

Рис. 2. Процесс взаимодействия привода с ПК.

Ограничения протокола обмена

В связи с ограниченностью внутренних буферов обмена необходимо учитывать максимальный размер прикладных пакетов, отправленных в устройство по UDP или TCP протоколам.

Максимальный размер пакета с прикладными данными, передаваемого по TCP соединению, составляет 710 байт.

Максимальный размер пакета с прикладными данными, передаваемого по протоколу UDP, составляет 1450 байт.

ПРИМЕЧАНИЕ:
Описанные ограничения также касаются сообщений, отправленных в широковещательном режиме.

В случае превышения указанных лимитов устройство может работать нестабильно.

Протокол ETH2CAN

Протокол ETH2CAN предназначен для обмена данными между ПК и ECG. Основная цель протокола выполнить ретрансляцию данных между сетями Ethernet и CAN.

Структура сообщения протокола приведена в Табл. 6.

Табл. 6. Формат протокола ETH2CAN.

Поле	Команда	Тело сообщения	Контрольная сумма (КС)
Размер,	1	От 0 до 64	1

байт			

Контрольная сумма

КС предназначена для подтверждения истинности данных, полученных по протоколу ETH2CAN.

КС представляет собой исключающее ИЛИ всех байт сообщения до применения механизма прозрачности.

Лист. 2. Листинг формирования контрольной суммы.

```
crc=0;
for(i=0;i<6; i++) crc ^= buf[i];
```

Команды протокола ETH2CAN

Описание команд протокола ETH2CAN представлены в Табл. 7.

Табл. 7. Описание команд протокола Ethernet-CAN.

Условное обозначение	Команда	Описание
HEARTBIT_ETH2CAN_PROTOCOL_CMD	0x09	Сердцебиение ECG. Передается шлюзом для проверки связи с частотой 1Гц
GETMODE_ETH2CAN_PROTOCOL_CMD	0x10	Запрос режима шлюза
MYMODE_ETH2CAN_PROTOCOL_CMD	0x11	Ответ шлюза с указанием режима
TOUART_ETH2CAN_PROTOCOL_CMD	0x14	Команда ретрансляции в UART (только для взаимодействия с пультом встроенного в состав привода СПС).
FROMUART_ETH2CAN_PROTOCOL_CMD	0x15	Команда ретрансляции из UART (только для взаимодействия с пультом встроенного в состав привода СПС)
TOCAN_ETH2CAN_PROTOCOL_CMD	0x16	Команда ретрансляции в CAN
FROMCAN_ETH2CAN_PROTOCOL_CMD	0x17	Команда ретрансляции из CAN

Команды разделены на две группы. Первая группа предназначена для взаимодействия непосредственно с устройством, вторая – для ретрансляции сообщений между интерфейсами Ethernet и CAN (UART).

Команда 0x09 – Сердцебиение

Номер параметра	Описание
0x09	Сердцебиение ECG. Передается шлюзом для проверки связи с частотой 1Гц

Команда передается без параметров.

Команда 0x10 – Запрос режима шлюза

Команда передается без параметров.

Данная команда отправляется из компьютера в широковещательном режиме с целью поиска всех ECG в локальной сети.

Команда 0x11 –Режим работы шлюза

Номер параметра	Кол-во байт	Описание
0x11	1	Режим работы устройства. 0 – Неизвестный режим.

		1 – Режим СПС. 2 - Режим Ethernet-CAN шлюза.

{UNKNOWN_MODE=0, SPS_MODE=1, ROUTER_MODE=2};

Устройство Ethernet-CAN шлюза основано на универсальном контроллере, который может входить в различные устройства. Данный контроллер, например, входит в состав сервопривода СПС на базе синхронного двигателя. В этом режиме протокол взаимодействия имеет другой формат. Протокол Ethernet-CAN ретрансляции относится только к устройствам, у которых параметр 0x11 имеет значение 2.

Команды 0x14/0x15 – Команды ретрансляции ETH2UART

В качестве данных передается команда технологического протокола, описанного в документе «***Описание параметров сервоприводов серии СПШ и СПС***».

Пример запроса параметра st4 «Тип привода» у привода серии СПС приведен в Лист. 3.

Лист. 3. Пример формирования команды передачи данных из ПК в контроллер привода СПС

```
//#define PARM_REQ_CMD            0x05   //Technological level commands
//#define PARM_VAL_CMD            0x06
//#define PARM_SET_CMD            0x07
//#define CAN_TRANSLATE_REQ_COM   0x0D   //CAN level commands
//#define TOCAN_ETH2CAN_PROTOCOL_CMD    0x16   //ECG level commands

size=0;
buf[size++] = TOUART_ETH2CAN_PROTOCOL_CMD;
buf[size++] = PARM_REQ_CMD;//Команда запроса параметра по технолог. протоколу
buf[size++] = 0x00;
buf[size++] = 0x46;
for(i=0; i<size; i++) buf[size] ^= buf[i]; //Подсчет КС
size++;
if(-1 == motors[0]->Send(buf, size) ) {
      motors.Reset();
}
```

Команды 0x16/0x17 – Команды ретрансляции ETH2CAN

В качестве данных передается команда протокола CAN.

Табл. 8. Данные команды протокола CAN

Размер, байт	4	1	От 0 до 8
Данные	CAN ID	Длина поля данных	CAN данные

Идентификатор CAN ID может быть 11 битным или 29 битным (см. п. 1.3). Подробнее об организации CAN сообщения можно ознакомиться в документе «***CAN Specification 2.0***».

Пример запроса параметра st4 «Тип привода» у привода серии СПШ приведен в Лист. 4.

Лист. 4. Пример формирования команды передачи данных из ПК в контроллер привода СПШ.

```
#define CAN_TRANSLATE_REQ_COM           0x0D
buf[0] = TOCAN_ETH2CAN_PROTOCOL_CMD;
if(interface_settings.can_address > 7) {
      //Формирование идентификатора CAN ID 29 бит
      id =         0x80000000 |
(CAN_TRANSLATE_REQ_COM<<14) |
(0<<7) |                                //Адрес ПК
(interface_settings.can_address << 0);  //Адрес СПШ
}
else {
      //Формирование идентификатора CAN ID 11 бит
      id = 0x00000000 |
(CAN_TRANSLATE_REQ_COM<<6) |
```

```
        (0<<3) |                                        //Адрес ПК
        (interface_settings.can_address << 0);          //Адрес СПШ
        }
        buf[1] = id;
        buf[2] = (id>>8);
        buf[3] = (id>>16);
        buf[4] = (id>>24);
        buf[5] = 3; //Кол-во байт данных
        buf[6] = 0x05;
        buf[7] = 0x00;
        buf[8] = 0x46;
        buf[9]=0;//CRC
        for(j=0;j<8;j++) buf[9] ^= buf[j];
        if(-1 == motors[0]->Send(buf, 10) ) {
              motors.Reset();
        }
```

Лист. 5. Пример процедуры подключения и передачи данных.

```
void send_param_example(void)
{
        int i, msg_len, len;
        unsigned char canaddr, cr;
        unsigned char trans_buf[16], b[32];
        SPSSocket* sps_socket;
        unsigned int id;

        //#define PARM_REQ_CMD            0x05   //Technological level commands
        //#define PARM_VAL_CMD            0x06
        //#define PARM_SET_CMD            0x07
        //#define CAN_TRANSLATE_REQ_COM   0x0D   //CAN level commands
        //#define TOCAN_ETH2CAN_PROTOCOL_CMD    0x16   //ECG level commands

        sps_socket = new SPSSocket((SPSDataConsumer*)&on_callback);
        sps_socket->Connect("192.168.2.25", 50023);

        //trans_buf[0] = PARM_REQ_CMD;                  //Technological protocol command
        //trans_buf[1] = 0x00;                          //Get parameter 0x00
        //trans_buf[2] = 0x15;
        //len = 3;
        trans_buf[0] = PARM_SET_CMD;
        trans_buf[1] = 0x02;                            //set parameter 0x0204
        trans_buf[2] = 0x04;
        trans_buf[3] = 0x00;                            //param's value
        trans_buf[4] = 0x04;
        trans_buf[5] = 0x00;
        trans_buf[6] = 0x00;
        len = 7;

        canaddr=0x01;                                   //Motor's CAN address
        id=canaddr;
        if(canaddr>7) {
              id |= 0x80000000;                         //ext id flag = 1
              id |= (CAN_TRANSLATE_REQ_COM<<14);        //CAN level protocol command
        }
        else {
              id |= (CAN_TRANSLATE_REQ_COM<<6);
        }
        b[0] = TOCAN_ETH2CAN_PROTOCOL_CMD;      //ECG router protocol command
        b[1] = id;
        b[2] = (id>>8);
        b[3] = (id>>16);
        b[4] = (id>>24);
        b[5]=len;
        for(i = 0; i < len; i++) b[6+i]=trans_buf[i];
        msg_len = i+6;

        cr=0;                                           //calculate CRC
        for(i = 0 ; i < msg_len; i++) {
```

```
            cr ^= b[i];
        }
        b[msg_len]=cr; msg_len++;
        if(NULL != sps_socket) sps_socket->Send(b, msg_len);

        Sleep(10);

        sps_socket->Close();
        delete sps_socket;
    }
```

Программные средства для работы с устройством ECG

Программа МотоМастер поддерживает данный протокол и выполняет автоматический поиск все доступные устройства ECG в сетях, к которым подключен данный компьютер.

В комплекте устройства ECG поставляется утилита ETH2CAN_terminal. Утилита позволяет взаимодействовать с устройством ECG.

Программа ETH2CAN_terminal имеет следующие команды:

-help Показать справку по программе.

-find Команда поиска всех устройств ECG, СПС, подключенных к ПК. Команда выполняет поиск в широковещательном режиме во всех подсетях, к которым подключен данный ПК.

-connect Подключение к ECG шлюзу.
Пример использования команды:

```
-connect 192.168.2.25
```

-disconnect Команда разрыва текущего TCP соединения. Команда без параметров.

-usb_mode Установить режим ретрансляции данных на локальную линию данных. В данном режиме все данные полученные по соединению TCP\IP будут переданы устройству, которые расположены на внутренней линии связи. В качестве такого приемника выступает центральный процессор привода СПС, в состав которого входит встроенный шлюз ECG.

-can_mode Установить режим ретрансляции данных на шину CAN.
Данный режим установлен в шлюзе по умолчанию.

-setid Установить адрес привода, к которому будут отправлены все запросы при вызове команд -send –get, -set, -startpwm, -stoppwm.
Команда имеет один параметр – адрес удаленного устройства.
При установке адреса 7 сообщения будут рассылаться в широковещательном режиме.
Пример использования команды:

```
-setid 0x01
```

-send Пересылка сообщения CAN приводу, установленного командой –setid.
Количество аргументов от 0 до 8.

-get Получить значение параметра.
Команда имеет один аргумент - адрес параметра.

Пример использования команды:

```
-get 0x040F
```

В данном примере выполняется запрос значения параметра dd11- Статус привода.

-set 0x0001 20 Установить значение параметра.
Команда имеет 2 аргумента - адрес параметра и значение.
Пример использования команды:

```
-set 0x0202 409600
```

Установить параметр ct2 – Задание скорости в значение 100 (в формате IQ12).

-startpwm Разрешить генерацию ШИМ.
Команда без параметров.

-stoppwm Запретить генерацию ШИМ.
Команда без параметров.

-sends, -sende Передача сообщения CAN со стандартным (11 бит) и расширенным (29 бит) идентификаторами.
Пример использования:
`-sends 0x341: 0x05 0x04 0x0F` Отправка в привод СПШ/СПС с CAN адресом 1 запроса значения параметра dd11.
`-sende 0x34008: 0x05 0x04 0x0F` Отправка в привод СПШ/СПС с CAN адресом 8 запроса значения параметра dd11.

```
При получении любых сообщений утилита отображает их в командной строке.
```

Процесс обмена данными между ПК и сервоприводом СПС

Процесс обмена приведен на Рис. 3.

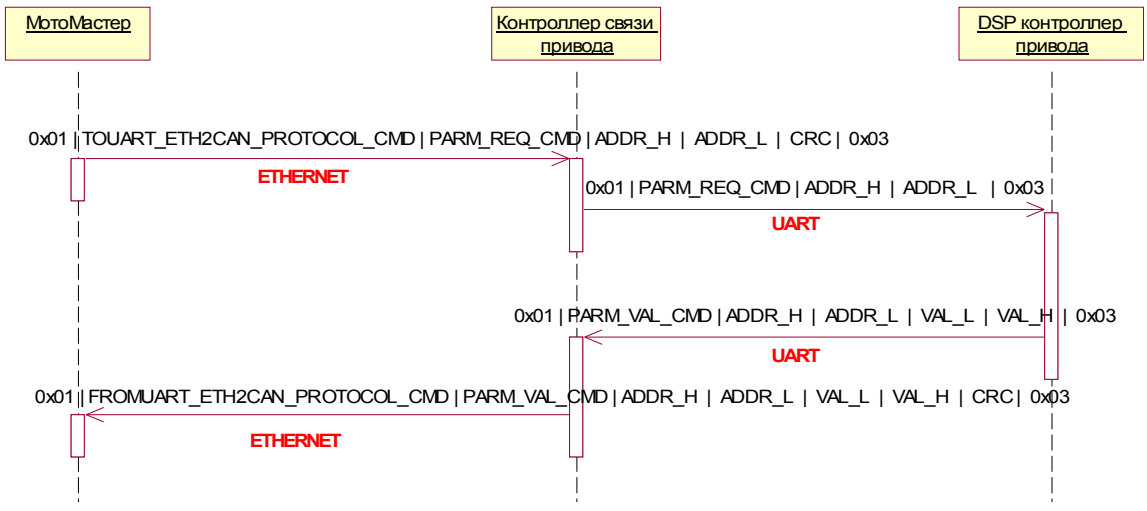

Рис. 3. Процесс обмена данными между ПК и сервоприводом СПС.

1.5. Взаимодействие с приводом по интерфейсу Ethernet через ECG шлюз

Процесс обмена с приводами серии СПШ можно выполнить с помощью Ethernet CAN шлюза (ECG). Данный шлюз выполняет ретрансляцию запросов и ответов между двумя сетями.

Процесс обмена выполняется по протоколу «***Протокол ETH2CAN***», описанного выше.

Процесс установки соединения аналогичен описанию в разделе «***Взаимодействие с приводом по интерфейсу Ethernet через встроенный контроллер связи (шлюз)***».

Процесс обмена похож на обмен с приводами СПС по Ethernet интерфейсу только в данном случае встроенный контроллер связи привода заменяется на ECG шлюз. При этом команды ретрансляции из/в UART заменяются на команды 0x16/0x17 – Команды ретрансляции ETH2CAN (Рис. 4).

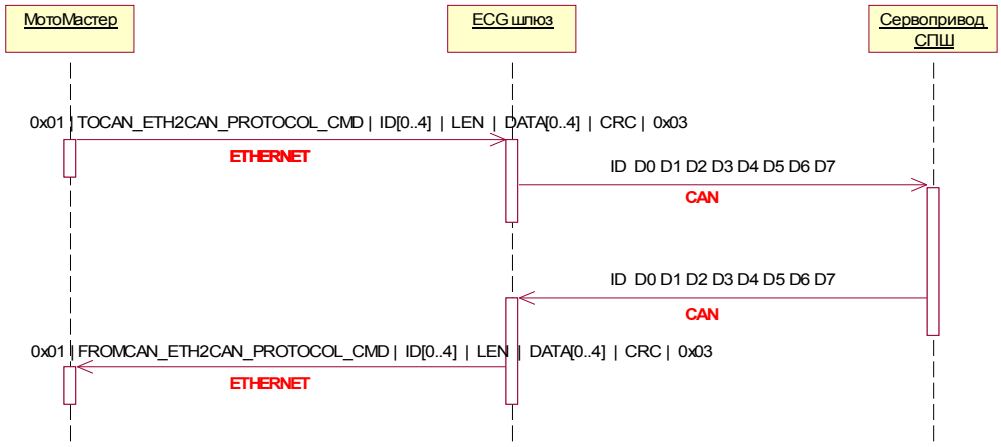

Рис. 4. Процесс обмена данными между ПК и сервоприводом СПШ.

Подробное описание работы со ECG шлюзом описано в документе «***ECG шлюз. Руководство по эксплуатации.pdf***».

1.6. Взаимодействие с приводом по интерфейсу CAN, ретранслированный через USB

Данный протокол позволяет обойтись без платы расширения CAN.

В данном случае функции платы расширения CAN выполняет один из приводов, который подключается к ПК в качестве ретранслятора.

Для реализации этого режима в приводе необходимо установить параметр “ip16: USB2CAN конвертор” в состояние «Вкл».

ВНИМАНИЕ! Не следует устанавливать параметр Ip16 в значение «Вкл» у более чем одного привода в сети CAN.

При этом процесс взаимодействия будет выглядеть так, как показано на Рис. 5.

```
#define PARM_USB2CAN_REQ_CMD      0x16
#define PARM_USB2CAN_FB_CMD       0x17
```

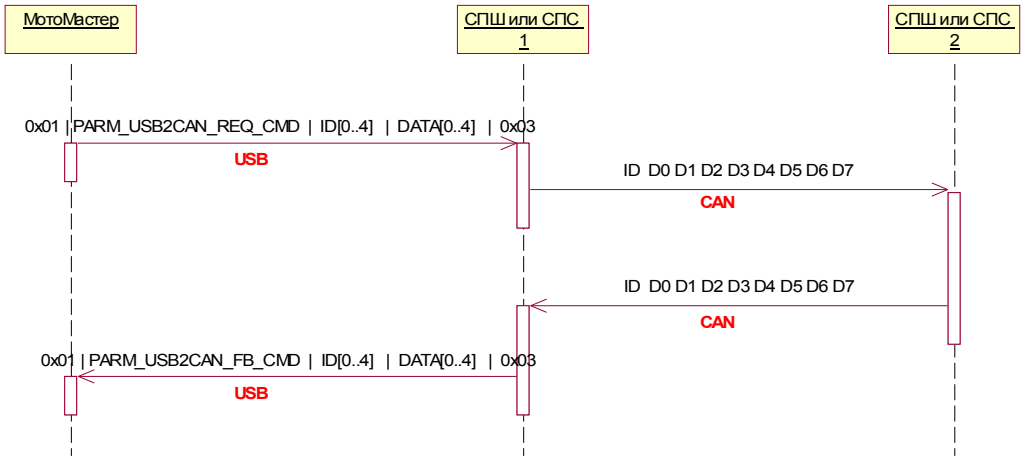

Рис. 5. Режим ретрансляции USB в CAN.

1.7. Интерфейс EtherCAT

EtherCAT – это гибкий сетевой протокол на базе сети Ethernet. Данная технология разработана специально для управления исполнительными устройствами. В сети EtherCAT должен быть один мастер. Количество одновременно подключенных ведомых устройств к одному контроллеру ограничено лишь теоретически достижимым числом 65535. Фактически же без проблем на физическом уровне можно управлять более сотни одновременно интерполируемых осей.

Интерфейс EtherCAT в первую очередь предназначен для контурного управления приводами в режиме реального времени. Данный тип взаимодействия выходит за рамки данного описания.

Однако в интерфейсе EtherCAT предусмотрена возможность передачи произвольных данных. Для этих целей предусмотрен протокол VoE (Vendor specific Over EtherCAT). В приводах серии СПС, укомплектованных интерфейсной платой EtherCAT, посредством обмена пакетами VoE имеется возможность взаимодействовать по технологическому протоколу.

Программа МотоМастер реализует данный протокол и умеет автоматически находить приводы с интерфейсом EtherCAT, которые подключены к компьютеру через Ethernet порт.

Для получения более детальной информации по вопросу взаимодействия по интерфейсу EtherCAT обращайтесь в отдел технической поддержки ЗАО «Сервотехника».