ОАО «ОТДЕЛЕНИЕ РАЗРАБОТКИ СИСТЕМ»

ПРОТОКОЛЫ ПЕРЕДАЧИ ДАННЫХ
при конфигурировании устройств на базе шины CAN

Формат сообщения CAN

Киров 2012

Формат сообщения CAN

1) 29-битный идентификатор, по нему производится арбитраж на шине CAN
2) 4 бита - поле длины сообщения, в котором можно задать значение от 0 до 8
3) поле данных – от 0 до 8 байт соответственно длине заданной в сообщении
4) контрольная сумма (формируется аппаратно)

В идентификаторе выделены 5 областей:
1) приоритет – 3 бита
2) резерв – 2 бита
3) адрес отправителя – 8 бит
4) адрес получателя – 8 бит
5) код сообщения – 8 бит

Приоритет – 0 – самый высокий, 7 – самый низкий, может задаваться пользователем, по умолчанию все устройства работают с приоритетом 4.

Адрес отправителя – может быть задан любым, кроме двух адресов 0x00 и 0xFF.

Адрес получателя – может быть задан любым.

Код сообщения – старший бит кода сообщения является типом сообщения 0 – команда, 1 ответ. В итоге доступными остаются 128 команд.

Адрес 0xFF – широковещательный адрес.

Адрес 0x00 – начальный адрес устройства, при подключении к линии “чистого” устройства у него будет установлен адрес 0x00.

Конфигурация устройства

Все разработанные устройства имеют двойную прошивку (загрузчик прошивок и пользовательская прошивка), и область конфигурации для каждой из прошивок.

Доступ к области конфигурации устройства осуществляется по команде 0x00-0x07 с определенным набором параметров расположенных в поле данных сообщения.

Переход между прошивками устройства осуществляется по команде 0x76, причем для перехода от загрузчика к пользовательской прошивке используется поле данных “UserProg”, для перехода обратно “Flasher”.

Механизм запуска

Любое устройство стартует с прошивки загрузчика и проверяет пользовательскую прошивку. Если точка входа в прошивку найдена и контрольная сумма прошивки совпадает с сохраненной, тогда запускается пользовательская прошивка. В противном случае запускается загрузчик.

Запуск загрузчика	Запуск пользовательской прошивки
(In) :1000FF81[08]0000003C50575231 (In) :1000FF81[07]466C6173686572	(In) :1000FF81[08]0000003C50575231

где

– (In) – направление передачи данных, In – команды поступающие из CAN сети, Out – команды уходящие в CAN сеть.

– 1000FF81 – идентификатор.

– [08] – длина поля данных.

– 0000003C – уникальный номер устройства.

– 50575231 – строка “PWR1” – означающая включение питания (перезагрузку) устройства.

– 466C6173686572 – строка “Flasher” – запущен загрузчик.

Идентификатор:

– 0x10 – 000**100**00b –100b=4 – приоритет, первые 000 – не используются (29 бит, а не 32), последние 00 – резерв.

– 0x00 – адрес отправителя (устройства).

– 0xFF – адрес получателя (для всех устройств).

– 0x81 – код ответа на команду 0x01.

Команда 0x00-0x07

Для доступа к данным конфигурации устройства может быть использован любой из 8-ми кодов команд (0x00 – 0x07).

Конфигурация хранится в 9 “массивах данных”:

– 0x00 – внутренний текст устройства, предполагается XML.

– 0x01 – массив констант неопределенного размера (размеры могут быть от 1, 2, 4 байта)

– 0x02 – массив байтовых переменных

– 0x03 – массив двухбайтовых переменных

– 0x04 – массив четырехбайтовых переменных

– 0x05 – массив слов, четырехбайтовый текст

– 0x06 – массив битовых переменных для прямого доступа к регистрам процессора

– 0x07 – массив описаний функций работы с данными устройства

– 0x08 – массив описаний функций конфигурирования устройства

В массивах (0x01-0x08) каждой переменной соответствуют две величины: значение и двухбайтовое описание переменной.

Определение размера массива:

Команда получения размера массива nn:

```
(Out):XXXXXXXX[01]nn
```

где:

– XXXXXXXX – идентификатор

– nn – однобайтовый номер массива

– каждая буква описывает одну шестнадцатеричную цифру, то есть для описания байта будет использовано 2 буквы.

Ответ: размер массива:

```
(In) :XXXXXXXX[02]ssss
```

где:

– ssss – двухбайтовый размер массива.

Получение значения переменной

Команда получения значения переменной №kkkk их массива №nn:

```
(Out):XXXXXXXX[03]nnkkkk
```

где:

– kkkk – двухбайтовый номер значения

Ответ – значение запрошенной переменной.

Пример №1: получение значения переменной №0 из массива №2

```
(Out):14E12000[03]020000
(In) :1020E180[01]20
```

Устройство сообщило значение этой переменной – 0x20

Пример №2: получение значения переменной №3 из массива №4

```
(Out):14E12000[03]040003
(In) :1020E180[04]000001F4
```

Устройство сообщило значение этой переменной – 0x000001F4

Пример №3: чтение внутреннего текста (массив №0)

```
(Out):14E12000[03]000000
(In) :1020E180[08]3C3F786D6C207665
(Out):14E12000[03]000008
(In) :1020E180[08]7273696F6E3D2231
(Out):14E12000[03]000004
(In) :1020E180[08]6C2076657273696F
```

Примечание:

– номер переменной для массива №0 определяет адрес во внутреннем тексте, причем чтение производится по 8 байт, таким образом, для полного чтения массива №0 требуется перебирать номера переменных через 8: 0, 8, 16, ….

Получение описания переменной

Получить описание текста из нулевого массива невозможно, да и не имеет смысла.

Команда получения описания переменной №kkkk их массива №nn:

```
(Out):XXXXXXXX[04]00kkknn
```

Ответ:

```
(In) :XXXXXXXX[02]tttt
```

Пример №1: получение описания переменной №0 из массива №2

```
(Out):14E12000[04]00000002
(In) :1020E180[02]0000
```

Устройство сообщило значение этой переменной – 0x0000

Пример №2: получение описания переменной №3 из массива №4

```
(Out):14E12000[04]00000304
(In) :1020E180[02]0005
```

Устройство сообщило значение этой переменной – 0x0005

Запись нового значения переменной

Возможно только для 5 массивов данных:

– 0x02 – массив байтовых переменных

– 0x03 – массив двухбайтовых переменных

– 0x04 – массив четырехбайтовых переменных

– 0x05 – массив слов, четырехбайтовый текст

– 0x06 – массив битовых переменных для прямого доступа к регистрам процессора

Общий формат команды записи значения vv переменной №kkkk в массив №nn:

```
(Out):XXXXXXXX[ss]nnkkkkvv
```

Общий формат ответа:

```
(In) :XXXXXXXX[cc]wwvv
```

где:

– ss и cc – размеры полей данных,

– nn – номер массива,

– kkkk – номер переменной массива,

– vv – новое значение переменной,

– ww – старое значение переменной.

Пример №1: запись массив №2, перед записью прочитаем значение переменной

```
(Out):14E12000[03]020000
(In) :1020E180[01]20
(Out):14E12000[04]02000019
(In) :1020E180[02]2019
```

Пример №2: запись массив №3, перед записью прочитаем значение переменной

```
(Out):14E12000[03]030003
(In) :1020E180[02]0007
(Out):14E12000[05]0300030001
(In) :1020E180[04]00070001
```

Пример №3: запись массив №4, перед записью прочитаем значение переменной

```
(Out):14E12000[03]040004
(In) :1020E180[04]00000004
(Out):14E12000[07]04000400000000
(In) :1020E180[08]0000000400000000
```

Пример №4: запись массив №5, перед записью прочитаем значение переменной

```
(Out):14E12000[03]050000
(In) :1020E180[04]444F3800
(Out):14E12000[07]05000030313233
(In) :1020E180[08]444F380030313233
```

Пример №5: запись массив №6, перед записью прочитаем значение переменной

```
(Out):14E12000[03]060001
(In) :1020E180[01]01
(Out):14E12000[04]06000100
(In) :1020E180[02]0100
(Out):14E12000[04]060001FF
(In) :1020E180[02]0001
```

Примечание:

– для 6-того массива установка бита в 1 производится командой 0xFF. Устройство может ответить любым числом от 0x00 до 0xFF, это число определяет бит или биты, которые были установлены в 1.

Широковещательный опрос (не рекомендуется использовать)

Команда 0x00 отправленная по адресу 0xFF (широковещательная), вызовет ответ всех устройств с запущенной пользовательской прошивкой.

Пример:

```
(Out):14E1FF00[00]
(In) :1022E180[08]0003041000000006
(In) :1020E180[08]0001010800000006
(In) :1021E180[08]0000011000000006
(In) :1023E180[08]00020E0400000006
(In) :1024E180[08]00020A1200000005
(In) :1025E180[08]0001010800000006
(In) :1026E180[08]00020E0400000006
(In) :1027E180[08]00020A0800000006
```

Ответило 8 устройств, ответ состоит из двух 4-х байтных слов, первое слово определяет тип прошивки, второе – определяет версию прошивки.

Тип устройства:

– первый байт (самый старший) – резервный байт

– второй байт – тип устройства (00 – DI, 01 – DO, 02 – AI, 03 – AO)

– третий байт – число бит на канал

– число каналов – число каналов

Многостраничная передача

Правила записи данных в пакеты:

1) дискретные данные с числом бит на канал равным 1, упаковываются в байты

2) аналоговые данные не упаковываются, то есть два 10-ти битных числа будут размещены в 4 байта (32 бита), а не в 3 (24 бита)

3) в случае если данные не могут разместится в одном пакете из 8 байт, первым байтом становится номер страницы, то есть максимальное количество страниц = 255 (255*7 = 1785 байт данных)

Пример многостраничной передачи:

```
(Out):05E12054[00]                  - запрос на получение массива из 8-ми
                                      аналоговых 10-ти битных данных
(In) :1020E1D4[07]00000500050008 - 00 - номер страницы;
                                      0005 - канал №7;
                                      0005 - канал №6;
                                      0008 - канал №5
(In) :1020E1D4[07]01000500080007 - 01 - номер страницы;
                                      0005 - канал №4;
                                      0008 - канал №3;
                                      0007 - канал №2
(In) :1020E1D4[05]02000A0009     - 02 - номер страницы;
                                      000A - канал №1;
                                      0009 - канал №0
```

Пример одностраничной передачи:

```
(Out):05E12058[00]                  - команда запроса на получение массива
                                      из 1-ого аналогового
                                      10-ти битного данного
(In) :1020E1D8[02]031C              - 031C - канал №0
```

Примечание:

– для стандартных функций чтения (**07.1XTT**) и записи (**07.2XTT**) тип передачи определяется автоматически, так как точно определяется количество данных требуемых для ответа на ту или иную команду.

– для остальных команд используется одностраничная передача, если иное не оговорено.

Назначение конфигурационных данных:

В дальнейшем при описании конфигурационных данных будем использовать формат:

`XX.TTTT`

где:

– XX – номер массива,

– TTTT – описание переменной массива.

Массив №1 – массив констант

01.0000 – точка входа в пользовательскую прошивку

01.0001 – контрольная сумма (простое суммирование) пользовательской прошивки

01.0002 – уникальный код устройства

01.0003 – контрольная сумма (CRC32) ядра CoreCAN

01.0004 – аппаратный адрес, если равен 0xFF – адрес задан программно, иначе значение является аппаратно заданным адресом.

01.0005 – версия базовой прошивки.

01.0006 – дата создания базовой прошивки.

01.0007 – версия пользовательской прошивки.

01.0008 – дата создания пользовательской прошивки.

01.0009 – текущее время, время работы после загрузки (uptime).

01.0010 – число пропущенных передач CAN-сообщений (происходит из-за перегруженности устройства)

01.0011 – число ошибок в CAN-шине

01.0012 – число событий WakeUp для модуля CAN устройства (в настоящее время эта функция модуля CAN не используется, должно быть равно 0).

01.0013 – число обработанных сообщений, за время работы устройства (с момента включения питания или перезагрузки).

01.0014 – адрес устройства компаньона.

01.100X – константа, описывающая тип массива, где X = 0x0...0xF - доступно всего 16 массивов.

Значение 4 байта: ttnnbbrr

tt - тип массива (00 - DI, 01 - DO, 02 - AI, 03 - AO, 04 – счетный вход, 05 – команда, 06 – текст UTF-16BE, 07 – образ массивов)

nn - число каналов (элементов массива)

bb - число бит на канал

rr - резерв (не используется, установлен в 00)

01.55AA – точка входа в прошивку загрузчика

01.Exxx – пользовательская конфигурация

Массив №2 – массив байтовых переменных

02.0000 – адрес устройства

Примечание:

– адрес будет изменен устройством после того как все команды поступившие в устройство будут обработаны, завершение изменения адреса подтверждается командой:

Пример: изменить адрес устройства №0x20 на №0x19

```
(Out):14E12000[04]02000019
(In) :1020E180[02]2019
(In) :1019FF81[08]416472203C2D3E19
```

где:

– 416472 – строка “Adr”

– 20 – старый адрес

– 3C2D3E – строка “<->”

– 19 – новый адрес

02.0001 – статус устройства (циклическая, спорадическая передача)

Примечание:

– нулевой бит (00000001b) – включение/выключение спорадической передачи

– первый бит (00000010b) – включение/выключение циклической передачи

02.0003 – значение регистра предделитель таймера A (trapre)

02.0004 – значение регистра таймера A (tra)

Примечание:

– таймер A задает единицу измерения переменной циклической передачи, timeout (**04.0005**).

– тактовая частота устройства $f_1 = 20\,000\,000\ Гц$.

– формула расчета единицы измерения переменной timeout:

$$time = \frac{8}{f_1} \cdot (trapre + 1) \cdot (tra + 1);\ time = \frac{8}{20\,000\,000} \cdot ([02.0004] + 1) \cdot ([02.0003] + 1).$$

– переменные trapre (02.0003) и tra (02.0004) байтовые и могут принимать значения в диапазоне 0x00…0xFF (или 0…255).

Пример: требуется установить единицу измерения timeout равную 1мс.

$$1мс = \frac{8}{20\,000\,000} \cdot ([02.0004]+1) \cdot ([02.0003]+1)$$

$$0{,}001 = 0{,}0000004 \cdot ([02.0004]+1) \cdot ([02.0003]+1)$$

$$2500 = ([02.0004]+1) \cdot ([02.0003]+1) \ (*)$$

Вывод: можно установить значения переменных в $[02.0003]=49$ и $[02.0004]=49$ или в $[02.0003]=124$ и $[02.0004]=19$ или в любое другое сочетание, удовлетворяющее уравнению (*).

02.0007 – тип универсальной команды, устарела, в настоящее время не используется

02.0008 – значение регистра предделитель таймера B (trbpre)

02.0009 – значение регистра таймера B (trb)

02.000A – скорость работы устройства по шине CAN, устанавливается в соответствии со следующей таблицей:

Значение	Скорость в бит/с	Параметры устанавливаемые в модуле CAN					
		fCAN	BRP	PTS	PBS1	PBS2	SJW
0	**1000000**	0	0	2	2	2	1
1	**500000**	0	0	6	5	5	3
2	**400000**	0	0	7	7	7	4
3	**250000**	2	0	2	2	2	1
4	**200000**	1	0	7	7	7	4
5	**125000**	3	0	2	2	2	1
6	**100000**	2	0	7	7	7	4
7	**80000**	0	4	7	7	7	4
8	**62500**	4	0	2	2	2	1
9	**50000**	3	0	7	7	7	4
10	**40000**	1	4	7	7	7	4
11	**31250**	4	1	2	2	2	1
12	**25000**	4	0	7	7	7	4
13	**20000**	2	4	7	7	7	4
14	**15625**	4	3	2	2	2	1
15	**12500**	4	4	2	2	2	1
16	**10000**	3	4	7	7	7	4

02.000B – тип управления сторожевым таймером выходов (1 – меандр, 2 – уровень 1, 3 – уровень 0).

02.Exxx – пользовательская конфигурация

Массив №3 – массив двухбайтовых переменных

03.0010 – переменная для настройки циклической передачи (каждый бит указывает на необходимость вернуть соответствующий массив)

03.0011 – переменная для настройки спорадической передачи (каждый бит указывает на необходимость вернуть соответствующий массив)

Правила:

1) Функции циклической и спорадической передачи нельзя вызвать

2) Команды-ответы приходящие от функций циклической и спорадической передачи содержат два байта данных

3) Каждый бит соответствует массиву данных, бит установленный в 1 говорит о том, что данные соответствующего массива были изменены

4) По измененному массиву автоматически генерируется команда-ответ простого чтения (**7.1X00**)

03.Exxx – пользовательская конфигурация

Массив №4 – массив четырехбайтовых переменных

04.0000 – тип устройства (устарела)

04.0001 – версия прошивки

04.0002 – дата создания прошивки

04.0005 – timeout – таймаут циклической передачи, по умолчанию в 1мс.

04.0006 – время самого длинного главного цикла в единицах измерения таймера B, по умолчанию в 100 мкс.

04.1XTT – переменные параметров массивов данных X, типа TT

TT=00 – фильтр дискретных данных.

TT=01 – апертура аналоговых данных (число квант, на которое требуется изменить значение аналогового сигнала относительно значения прошлой спорадической передачи, что бы сработала новая спорадическая передача).

Пример:

04.1000 – фильтр дискретных данных массива данных №0

04.1101 – апертура аналоговых данных массива данных №1

04.1301 – апертура аналоговых данных массива данных №3

TT=02 – блокировка канала, каждый бит отвечает за один канал (0 – заблокирован, 1 – в работе).

04.1202 = 0x1234 означает блокировку каналов №2, 4, 5, 9 и 12 массива данных №2.

TT=03 – инвертирование, каждый бит отвечает за один канал (0 – прямой, 1 – инверсный).

04.Exxx – пользовательская конфигурация

Массив №5 – массив слов

05.0000 – первые 4 символа названия устройства

05.0001 – вторые 4 символа названия устройства

05.0002 – третьи 4 символа названия устройства

Массив №6 – массив битовых переменных

06.0000 – tstart_tracr – бит запуска таймера A

06.0002 – tstart_trbcr – бит запуска таймера B

Примечание:

1) если возвращаемое значение отлично от нуля, то бит установлен.

2) устанавливать бит отправкой байта 0xFF.

```
(Out):14D70000[01]06        - прочитать количество элементов в массиве
(In) :1000D780[02]0002      - в массиве №6 хранится 2 элемента
(Out):14D70000[03]060000    - прочитать элемент №0000 из массива №6
(In) :1000D780[01]01        - элемент равен 01 – бит установлен
(Out):14D70000[04]06000000  - сбросить бит элемента №0000
(In) :1000D780[02]0100      - 01 – старое значение, 00 – новое
(Out):14D70000[04]060000FF  - установить бит элемента
(In) :1000D780[02]0001      - 00 – старое значение, 01 – новое
```

Массив №7 – описания функций работы с данными устройства

07.0010 – функция циклической передачи

07.0011 – функция спорадической передачи

07.1XTT – функция чтения массива X, типом TT

TT=00 – простое чтение

Пример использования команды такого типа:

```
(Out):05E12030[00]
(In) :1020E1B0[01]A5
```

TT=01 – зеркальное чтение (после перечисления всех элементов массива идет перечисление инверсных элементов массива, например 0xA5F0 - значение массива, тогда команда вернет 0xA5F05A0F)

Пример использования команды такого типа:

```
(Out):05E12031[00]
(In) :1020E1B1[02]A55A
```

TT=02 – чтение одного канала (Gx)

Пример использования команды такого типа:

```
(Out):05E12032[02]4700   - прочитать значение канала №0
(In) :1020E1B2[03]470001 - прочитанное значение канала №0 = 1
-----------------------------------
(Out):05E12032[02]4701   - прочитать значение канала №1
(In) :1020E1B2[03]470100 - прочитанное значение канала №1 = 0
```

Примечание код 47 - код символа G

TT=03 – заданной части одного канала (Gxp, где: ‘G’ – символ, x – номер канала, p – позиция)

Пример использования команды такого типа:

```
(Out):14E10558[03]470000           - прочитать данные канала №0 с
                                     нулевой позиции
(In) :1005E1D8[08]4700000622010104 – прочитаны данные: “0622010104”
(Out):14E10558[03]470001           - прочитать данные канала №0 с
                                     первой позиции
(In) :1005E1D8[07]47000122010104   - прочитаны данные: “22010104”
```

TT=04 – чтение текстов с указанной позиции (p, где: p – позиция)

Пример использования команды такого типа:

```
(Out):14D70311[01]00               - прочитать текст с позиции 0x00
(In) :1003D791[08]00123456789ABCDE – прочитан текст с позиции 0x00,
                                     данные: “123456789ABCDE” (7 байт)
(Out):14D70311[01]05               - прочитать текст с позиции 0x05
(In) :1003D791[08]05BCDE0000000000 – прочитан текст с позиции 0x05
                                     данные: “BCDE0000000000” (7 байт)
(Out):14D70311[01]4A               - если размер массива 80 символов и
                                     попытаться прочитать текст начиная
                                     с позиции 74 (0x4A), то устройство
                                     ответит:
(In) :1003D791[07]4A000000000000   - прочитан текст с позиции 0x4A,
                                     данные: “000000000000” (6 байт)
(Out):14D70311[01]4F               - если прочитать с позиции 79 (4F)
(In) :1003D791[02]4F00             - прочитает данные: “00” (1 байт)
(Out):14D70311[01]50               - на запрос данных адресов
                                     превышающих установленный в
                                     описании соответствующего массива
                                     (в данном случае 0x50) ответа не
                                     последует
```

TT=10 – чтение образа массивов, команда аналогична команде многостраничного простого чтения, но кроме байта номера страницы нулевым байтом передается номер образа. Формат поля данных при передаче:

0 байт: номер образа,

1 байт: номер фрагмента образа,

2-7 байты: данные образа.

Пример использования команды такого типа:

```
(Out):14E10060[00]                  - прочитать образ
(In) :1400E1E0[08]01003F07FFFFB8FF - образ №01, страница №00, данные
                                    - 3F07FFFFB8FF
(In) :1400E1E0[08]0101E1FFEFFFF9FF - 01, 01, E1FFEFFFF9FF
(In) :1400E1E0[08]0102E80028001B00 - 01, 02, E80028001B00
(In) :1400E1E0[08]01034C0016007E00 - 01, 03, 4C0016007E00
(In) :1400E1E0[08]0104B100D7000000 - 01, 04, B100D7000000
(In) :1400E1E0[08]0105000000000000 - 01, 05, 000000000000
(In) :1400E1E0[08]0106000000000000 - 01, 06, 000000000000
(In) :1400E1E0[06]010700FF0000     - 01, 07, 00FF0000
```

TT=11 – чтение описателя образа массивов, команда аналогична команде простого чтения, получает битовое поле с размером, указанным в типе массива.

Например, тип массива 01.1006 = 0x07190100 – 0x19 = 25 бит (4 байта).

```
(Out):14E10061[00]                  - запрос описателя образа
(In) :1400E1E1[04]23456701          - описатель образа 23 – младший байт
                                    - 00100011 – будут переданы 0, 1 и 5
                                    - объекты
```

07.2XTT – функция записи массива X, типом TT

TT=00 – простая

Пример использования команды такого типа:

```
(Out):05E12040[01]A7      - установить значение 0xA7
(In) :1020E1C0[01]A7      - значение 0xA7 установлено
```

TT=01 – зеркальная

Пример использования команды такого типа:

```
(Out):05E12041[02]A55A    - установить значение A5, 0xA5 XOR 0xFF = 0x5A
 (In) :1020E1C1[01]A5     - значение A5 установлено
-----------------------------------
(Out):05E12041[02]A5A5    - неправильная команда,
                            ответа на неё не последует
```

TT=02 – по XOR

Пример использования команды такого типа:

```
(Out):05E12042[01]0F      - инвертировать младший полубайт
                            данных устройства
(In) :1020E1C2[01]AA      - данные установлены в 0xAA
(Out):05E12042[01]0F      - инвертировать ещё раз
                            младший полубайт данных устройства
(In) :1020E1C2[01]A5      - данные установлены в 0xA5
```

TT=03 – по маске

Пример использования команды такого типа:

```
(Out):05E12040[01]00      - предварительно установим данные в 0x00
(In) :1020E1C0[01]00      - устройство подтвердило установку
(Out):05E12043[02]AAF0    - команда записи по маске (в данном случае 43),
                            в начале указывается маска, столько байт
                            сколько байт занимали бы данные, затем
                            идут сами данные
(In) :1020E1C3[01]A0      - устройство изменит биты *0*0*0*0
                            отмеченные * на значение 1_1_0_0_, в итоге
                            получим 10100000 или 0xA0
(Out):05E12043[02]55F0    - повторим команду записи по маске,
                            но с другой маской, обратной первой
(In) :1020E1C3[01]F0      - устройство изменит биты 0*0*0*0*
                            отмеченные * на значение _1_1_0_0, в итоге
                            получим 11110000 или 0xF0
```

TT=04 – одного канала (Sxy, где: ‘S’ – символ, x – номер канала, y – значение)

Пример использования команды такого типа:

```
(Out):05E12044[03]530100 - установить первый канал устройства в 0
                           53 – 'S', 01 – x, 00 - y
(In) :1020E1C4[01]A5     - данные установлены в 0xA5
(Out):05E12044[03]530101 - установить первый канал устройства в 1
(In) :1020E1C4[01]A7     - данные установлены в 0xA7
```

TT=05 – двухтактная, одного канала (Sx, Exy, где: ‘S’, ‘E’ – символы, определяющие фазу команды, x – номер канала, y – значение)

Пример использования команды такого типа:

```
(Out):05E12040[01]00     - установим вручную значение в массива в 0x00
(In) :1020E1C0[01]00     - устройство подтвердило установку в 0x00
(Out):05E12045[02]5303   - выберем командой Sx канал №3 - 0x03
(In) :1020E1C5[02]5303   - устройство подтвердило установку
(Out):05E12045[03]450301 - установим значение канала 0x03 в 0x01
(In) :1020E1C5[01]08     - данные установлены в 0x08
```

TT=06 – импульсная команда для одного канала (ttttxy, где: tttt – время импульса в тактах таймера B (100мкс по умолчанию), x – номер канала, y – значение), после завершения времени импульса канал устанавливается в значение обратное y.

Пример использования команды такого типа:

```
(Out):14E11326[06]0000000A0001 – установить на 1 мс канал №0 в 1
(In) :1013E1A6[06]0000000A0001 – команда принята, начато выполнение
```

TT=07 – заданной части одного канала (Sxpy, где: ‘S’ – символ, x – номер канала, p – позиция, y – значение)

Пример использования команды такого типа:

```
(Out):14E10559[08]5303020123456789 – записать данные "0123456789" в
                                     канал №3 в 2 позицию
(In) :1005E1D9[03]530302           - данные записаны в канал №3 в
                                     Позицию №2
```

TT=08 – запись текстов с указанной позиции (py, где: p – позиция, y – значение)

Пример использования команды такого типа:

```
(Out):14D70319[02]0012             - записать текст 0x12 в нулевую
                                     позицию
(In) :1003D799[02]0012             - записано в нулевую позицию 0x12
(Out):14D70319[08]00123456789ABCDE – записать 0x123456789ABCDE начиная
                                     с нулевой позиции
(In) :1003D799[08]00123456789ABCDE – записано успешно
(Out):14D70319[08]49123456789ABCDE – записать в позицию 0x49
(In) :1003D799[08]49123456789ABCDE – записана вся строка, успешно
(Out):14D70319[08]4A123456789ABCDE – если размер массива 80 символов и
                                     попытаться записать 7 символов
                                     начиная с позиции 74 (0x4A),
                                     то устройство ответит:
(In) :1003D799[07]4A123456789ABC   - записано 6 элементов
(Out):14D70319[01]00               - поле данных не указано, ответа нет
```

TT=11 – запись описателя образа массивов, команда аналогична команде простой записи, записывает битовое поле с размером, указанным в типе массива.

```
(Out):14E10062[00]23456801         - запись описателя образа
(In) :1400E1E2[04]23456801         - описатель образа записан
```

Массив №8 – описания функций конфигурирования устройства

08.0000 – служебная информация об устройстве, массивы переменных

08.F000 – перезагрузка по слову RESET

08.F001 – сохранение настроек в энергонезависимую память по слову FLASH

08.F002 – операции с реальным временем (4 байта, в мс)

Пример: запрос текущего времени у контроллера 0x12:

```
(Out):10E11272[00]
(In) :1012E1F2[04]05752310
```

Пример: установка текущего времени (устанавливает текущее время всем контроллерам, поддерживающим эту функциональность, ответа на команду установки времени не формируется):

```
(Out):10E1FF72[04]05752310
```

08.F006 – переход между прошивками по словам “UserProg” или “Flasher”. Для перехода из пользовательской прошивки в загрузчик требуется отправить слово “Flasher”.

Пример: переход в прошивку загрузчика из пользовательской прошивки:

```
(Out):14E12076[07]466C6173686572
(In) :1020FF81[08]0000003C50575231
(In) :1020FF81[07]466C6173686572
```

Пример: переход в пользовательскую прошивку из прошивки загрузчика:

```
(Out):14E12076[08]5573657250726F67
(In) :1020FF81[08]0000003C50575231
```

08.F007 – функция прошивки. понимает файлы *.mot (отправляется построчно, первые два байта отправляются байтами, далее по два байта воспринимается как число в 16-ричной системе счисления и отправляются байтом, в конце каждой строки отправляется пустая команда - "подтверждение выполнения", ответ на которую необходимо дождаться перед посылкой следующей строки), после прошивки рекомендуется вызвать функцию FLASH (**08.F001**)

Примечание:

– многостраничная передача не используется

– строка отправляется по 8 символов, в конце отправки строки передается “пустая” команда.

Пример: в файле test.mot записаны 5 строк:

```
S0030000FC
S2240048C0000098C30000FAC300008EC3000032C4000026C70000EAC900008EC3000094CA25
S2060048E00000D1
S20800FFFC9E5A00FF05
S804000000FB
```

Преобразуем строки к виду пригодному для отправки:

```
533030000FC
5332240048C0000098C30000FAC300008EC3000032C4000026C70000EAC900008EC3000094CA25
5332060048E00000D1
53320800FFFC9E5A00FF05
533804000000FB
```

Тогда диалог прошивки устройства №0x20 выглядит следующим образом:

1) отправляем первую строку

```
00:00:01.375 (Out):14E12077[06]5330030000FC
```

2) в конце строки добавляем “пустую” команду – выполнение отправленной строки

```
00:00:01.375 (Out):14E12077[00]
```

3) ждем ответа устройства

```
00:00:02.860 (In) :1020E1F7[08]4572616365204F6B
```

Примечание:

– команда S0 – очищает пользовательскую область

– если ответ – строка “Erace Ok” – очистка завершена успешно

– если ответ – строка“Error N” – очистка завершена с ошибкой, где N – символ кода ошибки.

4) отправляем вторую строку

```
00:00:02.860 (Out):14E12077[08]5332240048C00000
```

```
00:00:02.860 (Out):14E12077[08]98C30000FAC30000
00:00:02.860 (Out):14E12077[08]8EC3000032C40000
00:00:02.860 (Out):14E12077[08]26C70000EAC90000
00:00:02.860 (Out):14E12077[07]8EC3000094CA25
```

5) отправляем запрос на выполнение отправленной строки

```
00:00:02.860 (Out):14E12077[00]
```

6) ждем ответа устройства

```
00:00:02.860 (In) :1020E1F7[07]0048C020002D30
```

Примечание:

– команды S2 – записывают данные в пользовательскую область

– 0048C0 в данном случае – адрес по которому производится прошивка

– 20 – размер буфера данных, который требуется записать

– 00 – первый байт данных из буфера

– 2D – символ ‘–’

– 30 – код завершения операции прошивки (30 – прошивка завершена успешно, 31 – сравнение области ядра CoreCAN успешно, другие значения – ошибка)

7) отправляем третью строку

```
00:00:02.875 (Out):14E12077[08]5332060048E00000
00:00:02.875 (Out):14E12077[01]D1
```

8) отправляем запрос на выполнение отправленной строки

```
00:00:02.875 (Out):14E12077[00]
```

9) ждем ответа устройства

```
00:00:02.875 (In) :1020E1F7[07]0048E002002D30
```

10) отправляем предпоследнюю строку

```
00:00:02.875 (Out):14E12077[08]53320800FFFC9E5A
00:00:02.875 (Out):14E12077[03]00FF05
```

11) отправляем запрос на выполнение отправленной строки

```
00:00:02.875 (Out):14E12077[00]
```

12) ожидаем ответ

```
00:00:02.891 (In) :1020E1F7[08]53746172742D5A9E
```

Примечание:

– адрес фиксированной таблицы векторов из которой определяется точка входа в пользовательскую прошивку – 0x00FFFC, после записи в него устройство ответит строкой: “Start-AA”, где AA – два байта – точка входа в пользовательскую прошивку.

13) отправляем последнюю строку

```
00:00:02.891 (Out):14E12077[07]533804000000FB
```

14) отправляем запрос на выполнение отправленной строки

```
00:00:02.891 (Out):14E12077[00]
```

15) ожидаем ответ

```
00:00:02.922 (In) :1020E1F7[08]4352433D007F68D0
```

Примечание:

– ответ на команду S8 – строка “CRC=XXXX” – где XXXX – 4 байта код пользовательской прошивки, полученной простым суммированием пользовательской области.

Рекомендуется после прошивки выполнить последовательно команды: **08.F001** и **08.F000**. Первая команда – сохранит точку входа в пользовательскую прошивку и контрольную сумму пользовательской прошивки в энергонезависимую память. Вторая команда – перезагрузит устройство. В результате этих действий устройство произведет запуск пользовательской прошивки без явного перехода между прошивками.

Структуры

1) команда:

```
typedef struct
{
  unsigned char adr;      // адрес устройства назначения
  unsigned char code;     // код команды
  unsigned char mode;     // режим (00 – выключена, 01 – вкл.)
  unsigned char len;      // длина поля данных
  unsigned char data[8]; // поле данных
} s_extCommand;
```

В устройстве представляется как строка байт, размером от 4 до 12 байт (в зависимости от установленной длины). Прочитать возможно используя команду **`07.1X03`**, записать **`07.2x07`**.

Порядок передачи элементов:

Big-endian (BE) – старший младший, little-endian (LE) – младший старший.

При передаче последовательности элементов состоящих из нескольких бит следует говорить о порядке элементов в передаче и порядке битов в элементе.

Способы записи порядка передачи:

1) [порядок записи элементов]-[порядок записи бит в элементе]

2) [порядок записи бит в элементе]

– LE-BE – элементы передаются в порядке 0, 1, 2, …, N, а биты в элементе записаны k, k-1, …, 2, 1, 0. Характерно для передачи текстовых значений.

– BE-BE – элементы передаются в порядке N, N-1, …, 2, 1, 0, а биты в элементе записаны k, k-1, …, 2, 1, 0. Характерно для передачи измерений.

Порядок передачи элементов в функциях чтения/записи:

Номер функции	Тип функции			
	чтение		**запись**	
00	Простое	BE-BE	Простая	BE-BE
01	Зеркальное	BE-BE	Зеркальная	BE-BE
02	Одного канала	BE	По XOR	BE-BE
03	Заданной части	LE-BE	По маске (значение, маска)	BE-BE, BE-BE
04	Текста с позиции	LE-BE	Одного канала	BE
05			Двухтактная	BE
06			Импульсная (время, значение)	BE, BE
07			Заданной части одного канала	LE-BE
08			Текста с позиции	LE-BE