Протокол обновления прошивки модулей системы БЛОК  
по линии связи (CAN)

Протокол предназначен для внутрисистемного (без отчуждения) программирования модулей системы БЛОК по CAN-линии. Протокол должен соответствовать следующим требованиям:

1. Обеспечивать возможность программирования модуля по линии CAN без отключения его от системы
2. Обеспечивать надёжность доставки прошивки до модуля (исключать потерю и повреждение пакетов, обеспечивать повторную посылку данных при потере)
3. Обеспечивать возможность доставки бинарной прошивки для модулей без ОС и прошивки, представляющей набор программных файлов, для модулей с ОС
4. Обеспечить возможность проверки версии и целостности текущей прошивки в модуле
5. Исключить возможность прошивки модуля неродным ПО
6. Обеспечивать возможность передачи информации о параметрах модуля, прошивки и загрузчика

# Термины и определения

Ячейка – аппаратный узел системы. В случае, если система состоит из нескольких полукомплектов (каналов), каждый полукомплект (канал) считается отдельной ячейкой.

Модуль – функционально обособленный логический узел в системе. В одной ячейке может содержаться несколько программных модулей. Каждый модуль обладает своим идентификатором (номером), уникальным в пределах ячейки.

Прошивка (firmware) – программное обеспечение модуля. Может быть представлено как один бинарный исполняемый файл (для модулей без ОС), так и как каталог с файлами и подкаталогами (для модулей с ОС).

Программатор – программа, осуществляющая обновления прошивки в модуле.

Загрузчик – программа, исполняемая в модуле помимо основной прошивки, или функционально обособленная часть прошивки, отвечающая за реализацию данного протокола.

# OSI-модель протокола

Протокол прошивки состоит из нескольких уровней, согласно модели OSI, расположить которые можно следующим образом:

|  |  |
| --- | --- |
| Слой в модели OSI | Протокол |
| 5. Данные | Протокол FUDP |
| 4. Транспортный | ISO-TP (ISO 15765-2) |
| 3. Сетевой |
| 2. Канальный | CAN |
| 1. Физический | --- |

# FUDP over CAN

## CAN-сообщения

|  |  |  |
| --- | --- | --- |
| Дескриптор | Название | Назначение |
| FC08 | FU\_INIT | Запрос на перевод модуля в режим программирования |
| FC28 | FU\_PROG | ISO-TP сообщения, посылаемые программатором |
| FC48 | FU\_DEV | ISO-TP сообщения, посылаемые загрузчиком |

## ISO-TP

Для организации возможности передачи информации длиной более 8 байт и контроля потока поверх CAN используется протокол транспортного уровня ISO 15765-2 (ISO-TP). Применяется выравнивание CAN-фрейма до 8 байт.

# Протокол FUDP (Firmware Update Protocol)

Протокол FUDP работает поверх протокола ISO-TP.

Программатор является ведущим: он начинает обмен с отправки команды и дожидается ответа от загрузчика. Если через 2 секунды с момента окончания передачи команды не начато получение регламентируемого протоколом ответа, то программатор посылает команду на завершение сеанса программирования и начинает переход в новый сеанс. После 10 неудачных сеансов программирование считается не выполненным.

## Формат пакета FUDP

Пакет протокола FUDP состоит из заголовка и данных. Заголовком является первый байт ISO-TP сообщения. Он содержит идентификатор сообщения, определяющий тип сообщения. Остальные байты - данные сообщения, специфичные для каждого типа.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения* |
| N | данные сообщения |

## Переход в режим программирования

Для перевода модуля в режим программирования, программатор отправляет сообщение PROG\_INIT (идентификатор протокола FUDP: 01h).

### Сообщение PROG\_INIT (01h)

Сообщение передаётся для перевода соответствующего модуля в режим программирования.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ INIT (01h)* |
| 1,5 | ID ячейки (старшим вперёд) |
| 0,5 | Модификация |
| 1 | Номер программного модуля |
| 0,5 | Номер канала |
| 2,5 | Серийный номер (старшим вперёд) |

Это сообщение является Single Frame ISO-TP сообщением. Соответствующее ему CAN-сообщение выглядит так:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Дескриптор | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| FC08 | 07h | 01h |  |  | |  |  |  |

Таблица 1. Формат CAN-сообщения, содержащего PROG\_INIT

При получении такого сообщения со значениями параметров, равными параметрам модуля, прошивка модуля должна закончить работу и передать управление загрузчику.

«0» в значении любого из параметров значит «любой». При этом, если остальные параметры равны параметрам модуля, пошивка не завершает работу, а отправляет ответное сообщение со своими параметрами. Формат ответного сообщения совпадает с сообщением PROG\_INIT.

### Ответ на широковещательный запрос активных модулей – сообщение PROG\_BCAST\_RESPONSE (14h)

Сообщение предаётся на запрос PROG\_INIT, содержащий в качестве некоторых параметров «0», при условии, что остальные параметры совпадают с параметрами модуля.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ BCAST\_RESPONSE (14h)* |
| 1,5 | ID ячейки (старшим вперёд) |
| 0,5 | Модификация |
| 1 | Номер программного модуля |
| 0,5 | Номер канала |
| 2,5 | Серийный номер (старшим вперёд) |

Таблица 3. Формат сообщения PROG\_READ\_RQ

## Инициализация соединения

После завершения работы прошивки (или при загрузке после включения питания модуля) загрузчик принимает управление на 500 мс и ожидает получения от программатора PROG\_INIT сообщения со всеми заполненными параметрами, равными параметрам модуля. Программатор передаёт сообщение PROG\_INIT с интервалом в 100 мс. Если в течение 500 мс сообщение PROG\_INIT загрузчиком не получено, то происходит проверка целостности прошивки и её загрузка.

### Сообщение PROG\_STATUS (02h)

После получения сообщения PROG\_INIT загрузчик отправляет программатору сообщение PROG\_STATUS (идентификатор 02h), содержащее набор значений словарных свойств. Т.е. пар ключ – значение: 1 байт на ключ и 4 байта на значение. Длина сообщения PROG\_STATUS определяется количество заданных свойств. Максимальное количество одновременно используемых свойств = 64. Подробнее об словаре свойств смотри ниже.

Факт получения программирующим устройством сообщения PROG\_STATUS считается началом соединения.

## Запрос списка файлов

### Сообщение запроса списка файлов PROG\_LIST\_RQ (03h)

Для запроса списка файлов на программируемом модуле, программатор отправляет сообщение PROG\_LIST\_RQ.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_LIST\_RQ (03H)* |

### Сообщение списка файлов PROG\_LIST (04h)

На запрос списка файлов загрузчик отвечает сообщением содержащим список файлов. В случае отсутствия файловой системы на программируемом устройстве, именем файла является адрес начала файла в памяти (в шестнадцатеричном виде).

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_LIST (04h)* |
| 1 | Длина имени файла 1 |
| N | Имя файла 1 в кодировке Win-1251 |
| 4 | Размер файла 1 (байт) |
| 4 | Контрольная сумма файла 1 |
| 1 | Длина имени файла 2 |
| … | … |

Таблица 2. Формат сообщения PROG\_LIST

## Чтение файла

### Запрос на чтение – сообщение PROG\_READ\_RQ (05h)

Программатор может запросить чтение заданной части любого файла из перечисленных в PROG\_LIST. Для этого отправляется сообщение PROG\_READ\_RQ (05h).

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ READ\_RQ (05h)* |
| 1 | Длина имени файла |
| N | Имя файла в кодировке Win-1251 |
| 4 | Смещение от начала файла (байт) |
| 4 | Размер считываемого блока (байт): [0 – 4000?] |

Таблица 3. Формат сообщения PROG\_READ\_RQ

### Чтение данных – сообщение PROG\_READ (06h)

На сообщение PROG\_READ\_RQ загрузчик отвечает сообщением PROG\_READ.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ READ (06h)* |
| 1 | Код ошибки:  0 – Чтение успешно  1 – Файл не найден  2 – Недопустимое смещение (выходит за границу файла)  3 – Ошибка чтения |
| N | Считанные данные.  N – запрошенный размер или размер оставшейся части файла |

Таблица 4. Формат сообщения PROG\_READ

## Удаление файла

### Команда на удаления файла – сообщение PROG\_RM (07h)

Для удаления файла и очистки занимаемой им области памяти программатор отправляет сообщение PROG\_RM (07h)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ RM (07h)* |
| 1 | Длина имени файла |
| N | Имя файла в кодировке Win-1251 |

Таблица 5. Формат сообщения PROG\_RM

### Подтверждение удаления файла - сообщение PROG\_RM\_ACK (08h)

На запрос RPOG\_RM загрузчик должно ответить сообщением PROG\_RM\_ACK.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ RM (08h)* |
| 1 | Код ошибки:  0 – Удаление успешно  1 – Файл не существует  2 – Ошибка удаления файла |

Таблица 6. Формат сообщения PROG\_RM\_ACK

### Команда на очистку памяти – сообщение PROG\_MR\_PROPER (0Dh)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_MR\_PROPER (0Dh)* |
| *4* | *Шифр безопасности (4E8A1439)* |

Таблица 7. Формат сообщения PROG\_MR\_PROPER (0Dh)

### Подтверждение очистки памяти – сообщение PROG\_MR\_PROPER\_ACK (0Eh)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_MR\_PROPER \_ACK (0Eh)* |

Таблица 8. Формат сообщения PROG\_MR\_PROPER\_ACK (0Eh)

## Создание файла

### Команда на создание файла – сообщение PROG\_CREATE (09h)

Перед началом записи в файл, файл нужно создать.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ CREATE (09h)* |
| 1 | Длина имени файла N |
| N | Имя файла в кодировке Win-1251  (или адрес начала файла для устройств без ОС) |
| 4 | Размер файла |

Таблица 9. Формат сообщения PROG\_CREATE

### Подтверждение создания файла – сообщение PROG\_CREATE\_ACK (0Ah)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_ CREATE\_ACK (0Ah)* |
| 1 | Код ошибки:  0 – Файл создан успешно  1 – Файл с таким именем уже существует  2 – Превышено максимальное количество файлов  3 – Недостаточно памяти  4 – Ошибка создания |

Таблица 10. Формат сообщения PROG\_CREATE\_ACK

## Запись в файл

### Команда на запись в файл – сообщение PROG\_WRITE (0Bh)

Запись может вестись только в файл, созданный в рамках текущего сеанса программирования.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_WRITE (0Bh)* |
| 1 | Длина имени файла N |
| N | Имя файла в кодировке Win-1251  (или адрес начала файла для устройств без ОС) |
| 4 | Смещение от начала файла (байт) |

Таблица 11. Формат сообщения PROG\_WRITE

# Словарь свойств

Загрузчик отвечает за хранение в модуле списка свойств, слогласно нижеизлагаемому словарю.

Свойства делятся на 3 категории:

* **общие свойства** - доступны для записи и задаются программирующим устройством.
* **постоянные свойства блока** - задаются единожды при прошивке модуля на заводе. Содержат идентификацию блока.
* **свойства загрузчика** - генерируются загрузчиком и описывают его версию и возможности.

|  |  |
| --- | --- |
| Ключ | Свойство |
|  | Общие свойства (0 - 127) |
| 1 | Версия |
| 2 | Подверсия |
| 3 | Дата последнего обновления ПО |
| 4 | Доработка (версия) |
| 5 | Дата доработки |
| 6 | Контрольная сумма прошивки целиком |
|  |  |
|  | Постоянные свойства блока (128 - 191) |
| 129 | ID ячейки |
| 130 | Номер программного модуля |
| 131 | Серийный номер блока |
| 132 | Дата производства (год\*100 + месяц) |
| 133 | Номер канала (полукомплекта): 1, 2, 3, ... |
| 134 | Модификация ячейки |
|  |  |
|  | Свойтсва загрузчика (192 - 255) |
| 192 | Вид |
| 193 | Версия загрузчика |
| 194 | Наличие файловой системы (0 - без ФС, 1 - с ФС) |
| 195 | Версия протокола, по которой работает загрузчик |
| 196 | Наиболее старая версия протокола, которую поддерживает загрузчик |
|  |  |

## Запись значения в словарь свойств

### Команда на запись ЗНАЧЕНИЯ В СЛОВАРЬ СВОЙСТВ – сообщение param\_SET\_rq (0fh)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PARAM\_SET\_RQ (0Fh)* |
| 1  4 | Ключ  Значение свойства |

Таблица 13. Формат сообщения PARAMS\_SET\_RQ

### ПОДТВЕРЖДЕНИЕ записИ ЗНАЧЕНИЯ В СЛОВАРЬ СВОЙСТВ – сообщение param\_SET\_ACK (10h)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PARAM\_SET\_ACK (10h)* |
| 1 | Код ошибки:  0 – Значение свойства записано успешно  1 – Свойство «только для чтения»  2 – Превышено максимальное количество свойств |

Таблица 14. Формат сообщения PARAMS\_SET\_ACK

## УДАЛЕНИЕ параметра ИЗ СЛОВАРЯ СВОЙСТВ

### Команда на УДАЛЕНИЕ параметра ИЗ СЛОВАРЯ СВОЙСТВ – сообщение param\_RM\_rq (11h)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PARAM\_RM\_RQ (11h)* |
| 1 | Ключ |

Таблица 15. Формат сообщения PARAM\_RM\_RQ

### ПОДТВЕРЖДЕНИЕ УДАЛЕНИЯ параметра ИЗ СЛОВАРЯ СВОЙСТВ – сообщение param\_RM\_ACK (12h)

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PARAM\_RM\_ACK (12h)* |
| 1 | Код ошибки:  0 – Параметр удален успешно  1 – Параметр «только для чтения»  2 – Параметр не найден |

Таблица 16. Формат сообщения PARAM\_RM\_ACK

## выход из режима программирования

### Команда на выход из режима программирования

По окончании сессии программатор отправляет команду на выход из режима программирования. При получении данной команды программируемое устройство сохраняет проверяет целостность полученной прошивки, сохраняет изменения и передаёт управление (загружает) прошивку.

|  |  |
| --- | --- |
| Длина | Содержимое |
| *1* | *идентификатор сообщения PROG\_SUBMIT (13h)* |
| 1 |  |

Таблица 15. Формат сообщения PARAM\_RM\_RQ