Единый формат хранения ПО модулей системы БЛОК

Предложение

Оглавление

[Цели 1](#_Toc377654026)

[Несколько программных модулей в одной ячейке 2](#_Toc377654027)

[Файл ПО модуля 2](#_Toc377654028)

[Загрузчик 2](#_Toc377654029)

[Программы для работы 3](#_Toc377654030)

[Для разработчика: Выпуск версии при помощи FmPack 3](#_Toc377654031)

[Для пусконаладчика: Обновление по CAN при помощи CanProg 3](#_Toc377654032)

[Для завода: Первоначальная прошивка 4](#_Toc377654033)

[Устройство SFP-файла 4](#_Toc377654034)

# Цели

1. Иметь единый унифицированный формат хранения и распространения ПО всех разновидностей модулей системы БЛОК. Этот формат должен быть достаточно гибким, чтобы быть пригодным как для ПО, помещаемого в различные микроконтроллеры, так и для ПО, представляющего собой программу для ОС.
2. В файл с ПО внедрить дополнительную информацию. Например, номер версии и указание, для какого модуля это ПО предназначено.
3. Объединить в одном файле ПО для всех полукомплектов и модификаций ячейки.
4. Обеспечить защиту целостности ПО от умышленных и неумышленных модификаций.\*

# Несколько программных модулей в одной ячейке

Для ясности всегда в дальнейшем используются эти понятия:

Ячейка – аппаратно обособленная часть комплекса БЛОК.

Модуль (Программный модуль) – логическая единица, выполняющая функционально обособленную часть алгоритма комплекса БЛОК.

В одной классической ячейке без ОС зачастую находится один программный модуль, поэтому понятия ячейки и модуля были тождественны. С появлением ячеек под управлением ОС стало возможным на одном процессоре одновременно запустить несколько независимых программ. И обновлять их независимо. Отсюда появилась необходимость в разделении понятий ячейка и модуль.

Таким образом, ПО существует только для модуля, но не для ячейки.

# Файл ПО модуля

ПО модуля распространяется в виде SFP‑файла.

Один **SFP-файл это:**

* Одна версия
* Один модуль
* Все полукомплекты
* Все модификации ячейки

Один SFP‑файл содержит одну версию программы для одного модуля в вариантах для каждого полукомплекта и каждой модификации ячейки с этим модулем.

Помимо файлов программы SFP‑файл содержит описание:

* Номер версии, подверсии и литерную метку
* Дату выпуска версии
* Текстовые комментарии разработчика[[1]](#footnote-1)\*
* Защитную хэш-сумму\*

# Загрузчик

В ячейке находится загрузчик и программные модули. Роль загрузчика в том, чтобы обеспечить возможность обновления программ по линии CAN. Программы выполняют возложенные на них алгоритмы БЛОК и имеют минимальные представления о загрузчике и процессе обновления.

**Ячейка**

**=**

**Загрузчик**

**+**

**Программные модули**

Если программа поддерживает переход в режим программирования, то начать обновление ПО можно не перезагружая ячейку. В противном случае потребуется перезагрузка, при которой загрузчик на короткое время берёт на себя управление ячейкой.

# Программы для работы

* **FmPack** – создание SFP-файла
* **CanProg** – обновление по CAN
* **FmBurn** – первоначальная прошивка программатором

### Для разработчика: Выпуск версии при помощи FmPack

Разработчик формирует SFP-файл на этапе выпуска версии ПО модуля при помощи программы FmPack.

FmPack позволяет:

* Выбрать из каких файлов состоит программа, или в каком месте памяти она должна быть расположена;
* При необходимости задать разные программы для разных полукомплектов или разных модификаций ячейки;
* Присвоить версию и подверсию, добавить литерное обозначение и текстовый комментарий;
* Необходимо указать для модуля какой ячейки предназначено данное ПО.

### Для пусконаладчика: Обновление по CAN при помощи CanProg

После выбора SFP-файла программа проверяет его целостность по хэш-сумме и отображает, для какого модуля он предназначен и какую версию содержит.

При программировании CanProg:

* Находит все подключенные модули, для которых предназначен выбранный SFP‑файл, и переводит их в режим программирования по очереди.
* Проверяет версию установленного в модуле ПО и обновляет ПО при несовпадении версий.
* Контролирует целостность записанного в модуль ПО и применяет изменения.

Если модуль поддерживает безопасное обновление, то при перебое питания в процессе обновления в модуле остаётся предыдущее ПО. В противном случае гарантированно остаётся возможность обновления по CAN, а повреждённое ПО в модуле не загружается.

### Для завода: Первоначальная прошивка

Первоначальная прошивка подразумевает программирование пустой ячейки. На этом этапе устанавливается загрузчик и, опционально, некоторая версия основной программы. Загрузчик записывается программатором или иным специфическим для данной ячейки образом.

Важным этапом первоначальной прошивки является конфигурирование загрузчика. В процессе конфигурирования заполняется информация, необходимая для дальнейшей идентификации ячейки: её тип, модификация, серийный номер и дата изготовления. Изменить эту информацию возможно только во время конфигурирования загрузчика.

Для ячеек, использующих микроконтроллеры AVR, существует программа FmBurn, упрощающая процесс первоначальной прошивки.

# Устройство SFP-файла

SFP-файл является упакованным файлом и содержит xml-метаинформацию о содержащемся в нём ПО и один или более компонентов ПО.

Компонентом называется сборка ПО, адаптированная для конкретной цели.

Целью называется конкретный модуль конкретного полукомплекта конкретной модификации ячейки.

Пример 1. Есть ячейка A с двумя полукомплектами AI и AII.

Если одна программа подходит для обоих полукомплектов (AI и AII), необходим один компонент, целями которого будут AI и AII.

Если программа написана так, что для каждого полукомплекта компилируется своя «сборка», то целью одного компонента будет AI, а целью другого компонента будет AII.

Пример 2. Есть ячейка B. У неё существуют две модификации B1 и B2. Ячейка двухполукомплектная: BI и BII. И в ней находится три программных модуля: B1, B2 и B3.

Если ПО модуля 1 одинаковое для обоих полукомплектов и не зависит от модификации ячейки, то у него будет один компонент с 4 цели: .

Если ПО модуля 2 различное для полукомплектов и способно работать только с модификацией 1 ячейки, то у него будет два компонента, у каждого по одной цели. Цель компонента 1 – , а цель компонента 2 – .

Если ПО модуля 3 одинаковое для обоих полукомплектов ячейки, но способно работать только с модификацией 2, то у него будет одна сборка с 2 целями: .

Компонент в SFP-файле представляет собой содержание папки, т.е. вложенные в неё папки и файлы любой степени вложенности.

Для устройств с файловой системой это означает, что корневая папка модуля и все файлы и подпапки в ней будет соответствовать содержимому папки компонента.

Для устройств без файловой системы (например, AVR-микроконтроллеры) принимаются следующие соглашения по распределению файлов по памяти:

1. Имя файла соответствует начальному адресу размещения файла в соответствующей памяти в шестнадцатеричном формате. Файл хранится "как есть", т.е. в "бинарном формате". Расширение файла не указывается. Например, файл с именем 100 и размером в 256 байт будет размещён в области адресов 100h–1FFh
2. Папка "f" соответствует flash-памяти микроконтроллера
3. Папка "e" соответствует EERPOM-памяти микроконтроллера (если она имеется)
4. Присутствие файлов в корневой папке компонента, папок второго уровня вложенности и файлов, именованных не по указанным правилам не допускается, и приведёт к нарушению процесса прошивки модуля

Пример 1. Если ПО является программой для ОС, то компонентом будет являться содержимое папки этого компонента - исполняемый файл и произвольное число библиотек и ресурсов в виде файлов, находящихся радом с исполняемым или в подпапках. То, какой из этих файлов является исполняемым и как его запускать, форматом SFP не описывается и является предметом соглашения между загрузчиком данной ячейки и исполняемой программой.

Пример 2. Если ПО является «прошивкой» для микроконтроллера. Допустим, эта «прошивка» состоит из двух частей, которые должны быть размещены во flash-памяти контроллера следующим образом: [0h – 100h] и [120h - …h], и 64 байт данных, которые должны быть размещены в eeprom памяти начиная с адреса 30h.

Тогда дерево каталога компонента будет выглядеть следующим образом:

f/0 – бинарный файл, размером 100 байт, будет размещён во flash по адресу 0h

f/100 – бинарный файл, который будет размещён во flash начиная с адреса 100h

e/30 – бинарный файл, размером 64 байта, будет размещён в eeprom по адресу 30h

Необходимо обратить внимание на то, что фалы не имеют расширения .bin, как это обычно бывает.

В xml-файле содержится следующая информация:

* Версия
* Подверсия
* Литерная метка
* Дата релиза
* Текстовый комментарий
* Хэш-сумма
* Список компонентов (для каждого компонента указан список целей)

1. \* Не реализовано [↑](#footnote-ref-1)