Единый формат хранения ПО модулей системы БЛОК

ВЕРСИЯ 1

Есть предпосылки.

# 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.

# Формат и содержание SFP-файла

SFP-файл является упакованным файлом и содержит xml-метаинформацию о прошивке и один или более компонент ПО.

Компонентом называется совокупность файлов программы модуля, адаптированных для одного или более целевых модулей.... {тут надо объяснить хорошо}

Каждый компонент предтсавляет из себя папку с файлами программы и ресурсов. Содержимое папки компонента будет полностью соответствовать содержимому програмного модуля {ага?} после завершения процесса программирования.

Для устройств с файловой системой это означает, что корневая папка и все файлы и подпапки в ней програмного модуля будет соответствовать папке 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" соответстует 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

Что внутри файла?

Компоненты и цели

Структура файлов для устройств с ФС

Структура файлов для устройство без ОС (папочки e/, f/, имя файла = адрес начала), картинка с примером (можно сделать скриншот дерева в проводнике)

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