Акционерное общество «Энфорс»
ИНН 9723248779 КПП 772301001 ОГРН 1257700080659
109380, г. Москва, вн.тер.г. Муниципальный округ Люблино,
ул Чагинская, дом 4, строение 13, помещение 14/4

Инженерное превосходство в каждом ватте
info@enforce.moscow | enforce.moscow

Утверждаю
Генеральный директор АО
«ЭНФОРС»
Гуреев Михаил Владимирович
_____/
«30» апреля 2026 г.

Программное обеспечение «Интеллектуальная система управления системой накопления и хранения электрической энергии на основе батарей аккумуляторных литий-ионных (тяговой батареей)»

Описание технических средств хранения исходного текста и объектного кода ПО

Документация на программное обеспечение

Правообладатель: Акционерное общество «ЭНФОРС»

Автор: Гуреев Михаил Владимирович

г. Москва, 2026

Содержание

1 Общие сведения

Настоящий документ содержит описание технических средств хранения исходного текста и объектного кода программного обеспечения, а также технических средств компиляции исходного текста в объектный код программного обеспечения.

Документ подготовлен для программы ЭВМ «Интеллектуальная система управления системой накопления и хранения электрической энергии на основе батарей аккумуляторных литий-ионных (тяговой батареей)». Правообладателем программы является Акционерное общество «ЭНФОРС». Автор программы: Гуреев Михаил Владимирович.

Программное обеспечение реализовано на языках C/C++ и предназначено для работы в составе встроенной системы управления тяговой литий-ионной батареей на микроконтроллерной платформе с ядром ARM. Целевая среда выполнения связана с FreeRTOS, а сборка и проверка исходных материалов выполняются на рабочих станциях или сервере разработки под управлением Windows либо Linux.

Компиляция исходного текста в объектный код выполняется на технических средствах, размещенных на территории Российской Федерации. Для сборки используются CMake, компилятор C/C++, кросс-компилятор для ARM и служебные утилиты инструментальной цепочки, включая objcopy.

2 Состав программных материалов, подлежащих хранению

К программным материалам, подлежащим хранению, относятся исходные тексты, заголовочные файлы, сборочные сценарии, конфигурационные файлы, сведения о версии сборки, а также объектный код и производные бинарные файлы, получаемые в результате компиляции.

В составе предоставленных материалов присутствуют сборочные файлы CMake, файл build_datetime.h, материалы с исходными текстами системных модулей и портируемых слоев, включая абстракции операционной системы, файловой системы, отладки, обработки строк, времени, ресурсов и сетевых компонентов.

Сборочная конфигурация использует CMakeLists.txt как основной сценарий сборки. В нем подключаются описания модулей MCU, Unilib, FatFs, EEFS, CycloneTCP, исходные файлы приложения и сетевые компоненты. В качестве технического имени сборочной цели в материалах используется EvseController; при подготовке заявочных материалов оно рассматривается как внутреннее имя сборочной цели программного проекта.

Таблица 1 - Состав основных программных материалов

Группа материалов	Пример состава	Назначение
Исходные тексты	Файлы C/C++, заголовочные файлы, системные модули	Хранение текста программы в машиночитаемой форме

Сборочные сценарии	CMakeLists.txt, подключаемые *.cmake и списки исходных файлов	Описание порядка компиляции и состава сборки
Конфигурационные сведения	build_datetime.h, сведения о версии и параметрах сборки	Фиксация даты, времени и параметров результата
Объектный код	ELF, BIN, HEX, объектные файлы каталога сборки	Использование результата компиляции для проверки и эксплуатации

3 Технические средства хранения исходного текста программы

Исходный текст программы хранится в электронной форме в виде файловой структуры проекта. Основным техническим средством хранения является накопитель персонального компьютера, сервера разработки или специализированного файлового хранилища, обеспечивающего сохранность файлов, разграничение доступа и резервное копирование.

Допускается хранение исходного текста на локальном накопителе, сетевом файловом ресурсе, в защищенном архиве ZIP, а также в системе контроля версий при условии ограничения доступа к репозиторию и сохранения истории изменений. Для долговременного хранения рекомендуется использовать дублирование архива на независимом носителе.

Исходный текст должен храниться в структуре каталогов, сохраняющей относительные пути к файлам сборки и подключаемым модулям. Недопустимо изменять расположение каталогов MCU, Unilib, fatfs, src, EEFS, CycloneTCP и связанных файловых списков без соответствующей корректировки сборочных сценариев.

4 Технические средства хранения объектного кода программы

Объектный код и производные файлы программы формируются в процессе сборки и хранятся в каталоге сборки, а также в выходном каталоге bin, создаваемом сборочным сценарием. В состав объектных и исполняемых результатов могут входить ELF-файл, бинарный файл BIN и файл Intel HEX.

Файл ELF используется как основной файл результата компоновки, содержащий машинный код и служебную информацию для отладки и анализа. Файлы BIN и HEX используются как производные представления объектного кода, применимые для записи во Flash-память целевого устройства или передачи в составе комплекта поставки.

Для архивного хранения объектного кода рекомендуется сохранять отдельный каталог с результатами сборки, содержащий файлы .elf, .bin, .hex, сведения о дате и времени сборки, версию компилятора, параметры CMake и контрольные суммы файлов. Объектный код должен храниться отдельно от

исходного текста либо в выделенном подкаталоге, чтобы исключить подмену исходных материалов результатами сборки.

5 Технические средства компиляции исходного текста в объектный код

Компиляция исходного текста в объектный код выполняется на персональном компьютере или сервере разработки под управлением операционной системы Windows или Linux. Минимальным обязательным средством организации сборки является CMake версии не ниже 3.1, что соответствует требованиям основного сборочного сценария.

В качестве компилятора используется совместимый компилятор C/C++. Для сборки под целевую микроконтроллерную платформу применяется кросс-компилятор для архитектуры ARM, например инструментальная цепочка на базе GCC ARM Embedded или иной совместимый комплект, содержащий компилятор, компоновщик и утилиту objcopy.

Утилита objcopy используется сборочным сценарием для преобразования результата компоновки в форматы BIN и Intel HEX. Сборочная система также использует переменные CMAKE_OBJCOPY и CMAKE_CURRENT_SOURCE_DIR, что позволяет выполнять постобработку результата сборки и помещать производные файлы в каталог bin.

Таблица 2 - Технические средства компиляции

Средство	**Минимальное требование**	**Назначение**
Рабочая станция или сервер	Windows или Linux	Размещение исходных материалов и запуск сборки
CMake	Версия не ниже 3.1	Генерация файлов сборки и управление проектом
Компилятор C/C++	Совместимый компилятор для целевой платформы	Компиляция исходного текста
Кросс-компилятор ARM	GCC ARM Embedded или совместимая инструментальная цепочка	Сборка объектного кода для микроконтроллерной платформы
objcopy	Утилита из состава инструментальной цепочки	Преобразование ELF в BIN и HEX

6 Порядок компиляции и формирования объектного кода

Перед компиляцией необходимо развернуть архив исходных материалов в рабочий каталог и убедиться, что сохранена структура каталогов проекта. Далее на рабочем месте устанавливаются CMake, компилятор C/C++, средства сборки и при необходимости кросс-компилятор для целевой платформы ARM.

Типовой порядок сборки включает конфигурирование проекта, генерацию файлов сборки, компиляцию исходных текстов, компоновку исполняемого образа и постобработку результата. После успешной сборки в каталоге bin должны быть сформированы файлы с расширениями .elf, .bin и .hex.

При сборке для экспертной проверки рекомендуется фиксировать версию инструментальной цепочки, параметры командной строки, дату и время сборки, а также контрольные суммы полученных файлов. Это позволяет воспроизвести результат и подтвердить соответствие объектного кода исходному тексту.

Пример последовательности команд для типовой проверки сборки:

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Release
cmake --build build --config Release
sha256sum bin/* > checksums.txt
```

7 Контроль целостности и воспроизводимость хранения

Для контроля целостности исходного текста и объектного кода рекомендуется формировать файл контрольных сумм по алгоритму SHA-256 или иному криптографически стойкому алгоритму. Контрольные суммы должны рассчитываться отдельно для архива исходных материалов и для каждого итогового файла объектного кода.

При передаче материалов на экспертизу целесообразно хранить исходные тексты, объектный код, документацию и файл контрольных сумм в едином архиве. Архив должен иметь устойчивое имя, включающее обозначение программы, дату формирования и при необходимости номер версии.

Для исключения несанкционированного изменения программных материалов архив должен храниться в режиме ограниченного доступа. Резервная копия архива размещается на независимом носителе или в защищенном файловом хранилище организации.

8 Рекомендуемая структура архива документации

Для передачи и хранения материалов рекомендуется использовать архив, содержащий отдельные каталоги для документации, исходного текста, объектного кода и контрольных сведений. Такая структура упрощает экспертную проверку и исключает смешение исходных и производных файлов.

В каталог Documentation помещаются документы, описывающие архитектуру, установку, функциональные характеристики, эксплуатацию, поддержку жизненного цикла, а также настоящий документ о технических средствах хранения и компиляции. В каталог Source помещаются исходные тексты и сборочные файлы. В каталог Build или ObjectCode помещаются результаты компиляции.

Файл checksums.txt или аналогичный файл должен содержать контрольные суммы ключевых материалов. Дополнительно может включаться файл README.txt с кратким описанием состава архива и порядка проверки.

9 Заключение

Описанные технические средства хранения и компиляции обеспечивают возможность сохранения исходного текста программы, получения объектного кода, архивного хранения результатов сборки и последующей экспертной проверки материалов.

Применение CMake, компилятора C/C++, кросс-компилятора для ARM и утилиты objcopy позволяет воспроизводимо формировать объектный код программы из исходных текстов при сохранении структуры проекта и фиксировании параметров сборки.

Для повышения надежности комплекта рекомендуется хранить исходный текст, объектный код, документацию и контрольные суммы в едином архиве с разграничением каталогов и резервным копированием.