Руководство пользователя

Программное обеспечение CMSIS-DAP на программатор
для микросхем с ядром CORTEX-M

Содержание

1. Основные возможности программы

CMSIS-DAP[1] – это программное обеспечение для программирования и отладки микроконтроллеров с ядром Cortex-M. CMSIS-DAP предназначен для работы с «Комплектом программатора для микросхем с ядром CORTEX-M ТСКЯ.468998.109», разработанным компанией АО «ПКК Миландр».

ПО программатора поддерживает два протокола обмена данными с ПК - CMSIS-DAP v1 и CMSIS-DAP v2. Протокол v1 использует USB-класс HID и является стабильной, полностью проверенной версией (**STABLE**, 1.0.3). Протокол v2 использует Bulk-передачу данных, что обеспечивает более высокую скорость программирования и отладки по сравнению с v1, и в настоящий момент находится в активной разработке (**BETA**, 2.0.0). Версия 2.0.0 полностью сохраняет обратную совместимость с протоколом v1. Переключение между версиями описано в разделе «Обновление ПО программатора» (Раздел 2.5).

Программирование FLASH-памяти:

- работа с основной и информационной (версия 1.5+) областями FLASH-памяти;
- полное стирание памяти;
- запись программы в микроконтроллер из файлов форматов HEX[3] и ELF[2];
- сохранение программы из микроконтроллера в файл формата HEX[3];
- сравнение записанной программы в микроконтроллер с данными из файлов форматов HEX[3] и ELF[2].

Отладочные возможности:

- загрузка программы в ОЗУ с последующим запуском;
- отладочный вывод с помощью интерфейса SWO с возможностью сохранения принятых данных в файл формата TXT;
- чтение и запись регистров ядра, переменных в памяти;
- формирование аппаратного сигнала RESET.

Дополнительные возможности:

- обновление встроенного программного обеспечения программатора;

[1] CMSIS-DAP доступен для скачивания по ссылке: https://ic.milandr.ru/upload/iblock/665/d8vms7yinqwlkex79ddsh04ga9af8cgr/CMSIS-DAPi.zip

[2] Только elf-файлы, полученные в средах IAR EWB и ARM KEIL

[3] Требования к используемому hex-файлу (неактуально для версий ПО 2.0.0 и выше):
- адреса должны быть отсортированы в порядке возрастания;
- максимальная длина строки в байтах – 16 байт.
Hex-файл, полученный в среде CodeMaster ARM, не удовлетворяет этим условиям, поэтому необходимо воспользоваться инструкцией https://support.milandr.ru/base/spravka/32-razryadnye-mikrokontrollery/sredy-otladki-i-programmatory/36885/

- возможность исполнения пользовательских скриптов перед чтением, записью и стиранием памяти;
- поддержка UART-загрузчика компании АО «ПКК Миландр».

> **⚠ Предупреждение**
>
> В процессе работы программатора (первой ревизии) аппаратный сброс микроконтроллера допускается производить исключительно средствами программатора. Другие способы могут привести к выходу из строя программатора.

1.1. Системные требования

Для работы программы CMSIS-DAP на ПК необходимо:

- операционная система Windows 7 SP1 или новее, разрядность 32 или 64 бита;
- свободный порт USB;
- права администратора на этапе установки программного обеспечения;
- не менее 100 МБ свободного места на жёстком диске.

2. Описание интерфейса программы CMSIS-DAP

Общий вид программы CMSIS-DAP приведен на рис. 1. Начиная с версии 2.0.0 программа поддерживает светлую и тёмную темы оформления, переключение между которыми выполняется ползунком в правом нижнем углу окна программы.

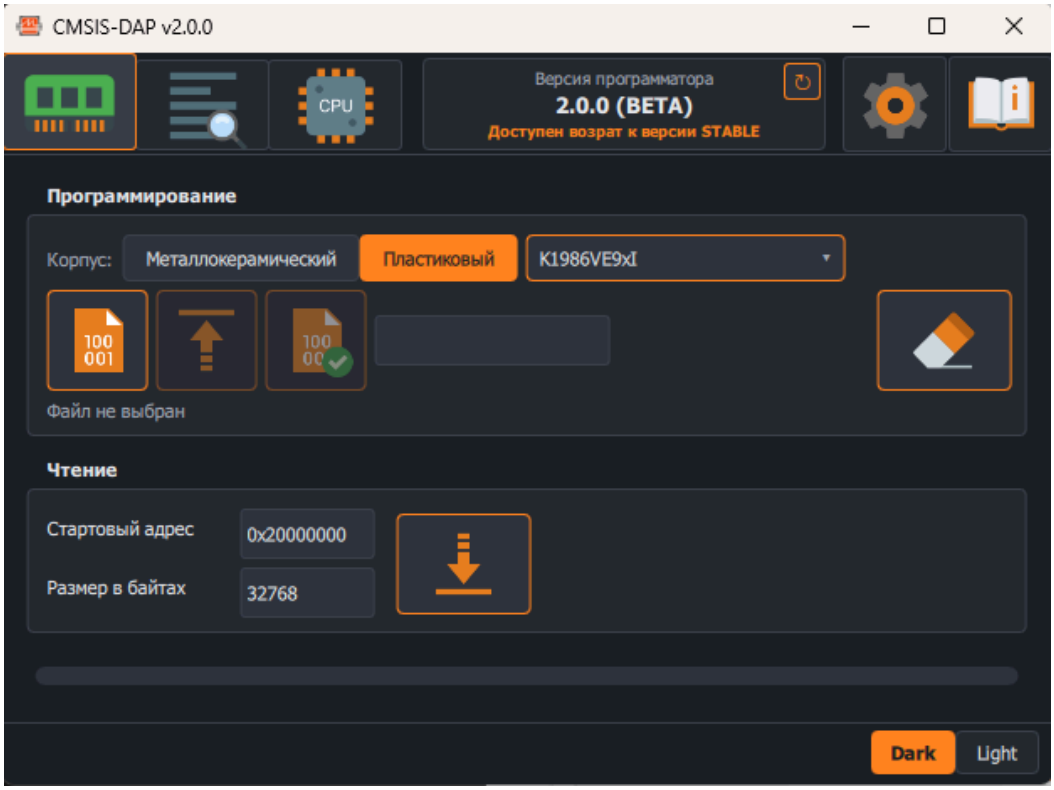

Рисунок 1 — Общий вид программы (тёмная тема)[4]

Программа содержит три вкладки:

1. вкладка программирования памяти;
2. вкладка отображения вывода данных SWO;
3. вкладка доступа к регистрам ядра и переменным в памяти.

Также в верхнем правом углу расположены кнопки вызова настроек и справки.

[4] Вид управляющего окна утилиты может отличаться в зависимости от версии утилиты

2.1. Вкладка программирования памяти

Первая вкладка программы CMSIS-DAP – «Программирование памяти» (рис. 2), которая позволяет осуществить работу с памятью микроконтроллера. Доступны операции записи, чтения, сравнения и стирания памяти. Операция стирания выполняет полное стирание FLASH-памяти микроконтроллера.

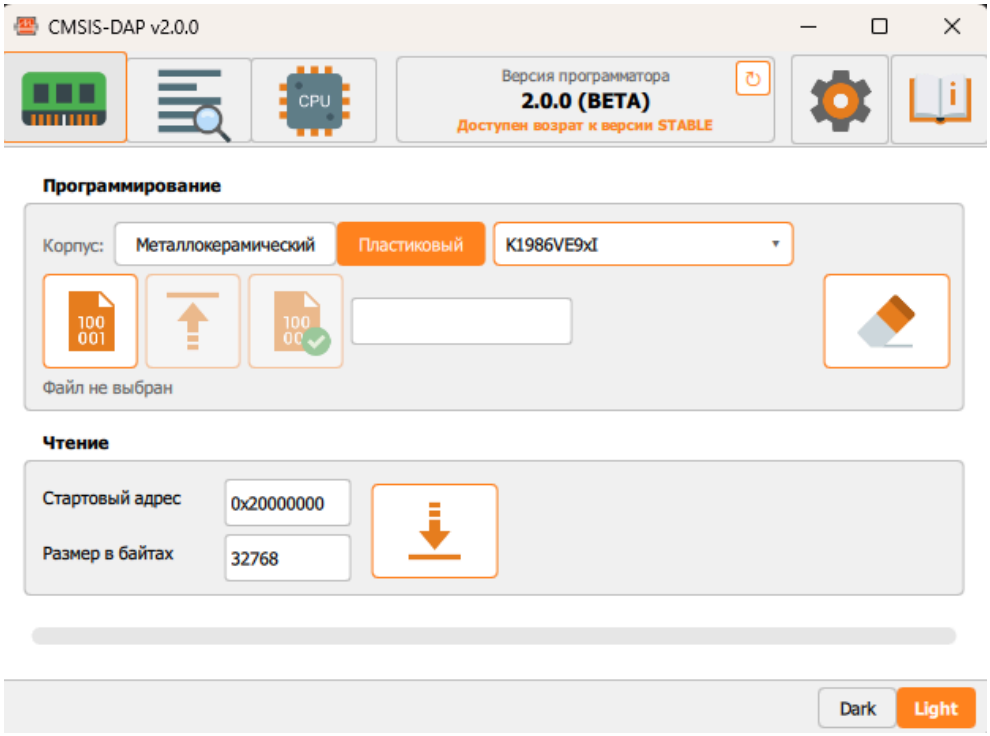

Рисунок 2 — Вкладка программирования памяти

> **Примечание**
>
> Работа с памятью недоступна, пока запущено отображение данных SWO.

2.1.1. Запись данных в память

Для загрузки данных необходимо (рис. 3):

1. переключателем выбрать тип корпуса микроконтроллера – «Металлокерамический» или «Пластиковый» (добавлено в версии 2.0.0). В зависимости от выбора активируется соответствующий выпадающий список загрузчиков;
2. выбрать из выпадающего списка необходимый загрузчик. Список доступных загрузчиков приведен в разделе «Встроенные загрузчики» (Раздел 5.2);
3. выбрать файл прошивки формата hex[3] или elf[2]. После выбора файла под кнопкой отобразится полный путь до файла и значение его CRC;
4. нажать на кнопку записи прошивки и дождаться завершения процедуры записи.

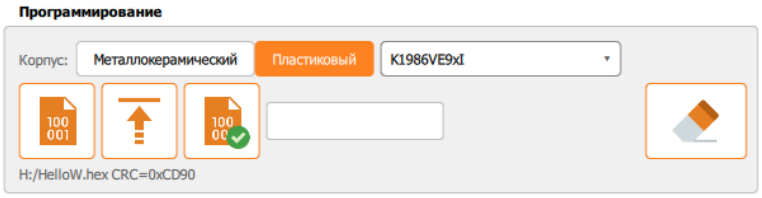

Рисунок 3 — Выбор файла для программирования

2.1.1.1. UART-загрузчик

Если загрузчик поддерживает возможность работы через UART, то можно задействовать данный режим, переключив ползунок. После этого можно задать требуемую скорость обмена (рис. 4).

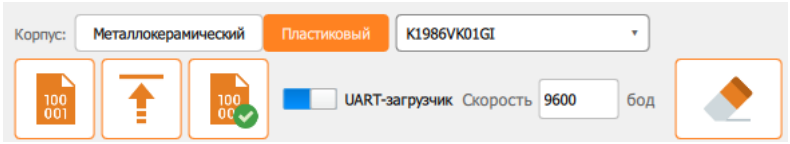

Рисунок 4 — Настройка UART-загрузчика

Далее необходимо выставить на микроконтроллере режим загрузки по UART и выполнить подключение, как показано в табл. 1.

Таблица 1 — Подключение интерфейса UART

Номер вывода программатора	Обозначение вывода микроконтроллера
17	TX
19	RX
15	Reset
1	Vcc

1. Соединить «земли».
2. Подключение вывода RESET не обязательно, но если он не подключен, перед использованием интерфейса UART необходимо самостоятельно осуществить сброс микроконтроллера[5].

> **⚠ Предупреждение**
>
> Функционал UART-загрузчика оставлен для совместимости с предыдущей версией CMSIS-DAP и в текущих загрузчиках не используется. С помощью UART-загрузчика возможен доступ к ОЗУ (запись/чтение), а также запуск программы на выполнение, при этом доступ к Flash-памяти не реализован.

[5] **Внимание:** в процессе работы программатора (первой ревизии) аппаратный сброс микроконтроллера допускается производить исключительно средствами программатора. Другие способы могут привести к выходу из строя программатора.

2.1.1.2. Дополнительные опции

Если загрузчик поддерживает дополнительные опции, то их можно ввести в поле рядом с кнопкой сверки данных (рис. 5).

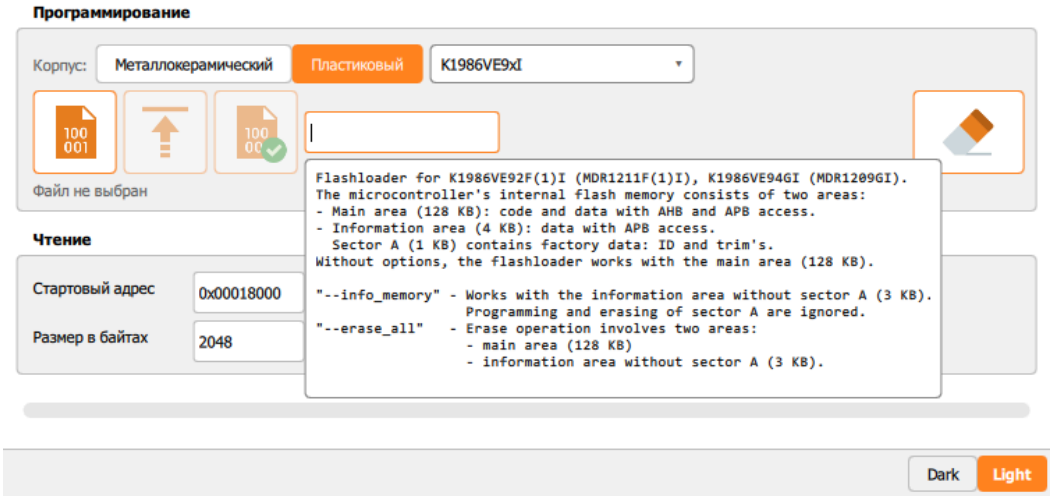

Рисунок 5 — Всплывающее окно дополнительных опций

При наведении курсора на данное поле появится подсказка с описанием доступных параметров. Список параметров для каждого встроенного загрузчика указан в разделе с описанием загрузчиков (Раздел 5.2).

2.1.2. Верификация запрограммированных данных

Для верификации данных необходимо:

1. выбрать файл прошивки формата hex[3] или elf[2]. После выбора файла под кнопкой отобразится полный путь до файла и значение его CRC;
2. нажать на кнопку запуска процедуры сверки данных и дождаться завершения процесса;
3. результат будет отображен под шкалой текущего прогресса выполнения. Также будет отображено CRC данных из памяти (рис. 6).

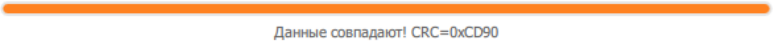

Рисунок 6 — CRC данных из памяти

2.1.3. Считывание данных в файл

Для считывания данных из памяти в файл формата Intel HEX необходимо (рис. 7):

1. указать стартовый адрес и размер загружаемых данных в байтах;
2. нажать на кнопку считывания данных в файл и задать имя файла для сохранения;
3. дождаться завершения операции. По завершении будет отображено количество прочитанных данных и значение их CRC.

Рисунок 7 — Считывание данных в файл

2.2. Вкладка отображения вывода данных SWO

Вторая вкладка – «Отображение вывода данных через SWO» (рис. 8), в которой отображаются данные, поступающие через отладочный вывод с помощью интерфейса SWO. Все принятые данные могут быть сохранены в текстовый файл (.txt).

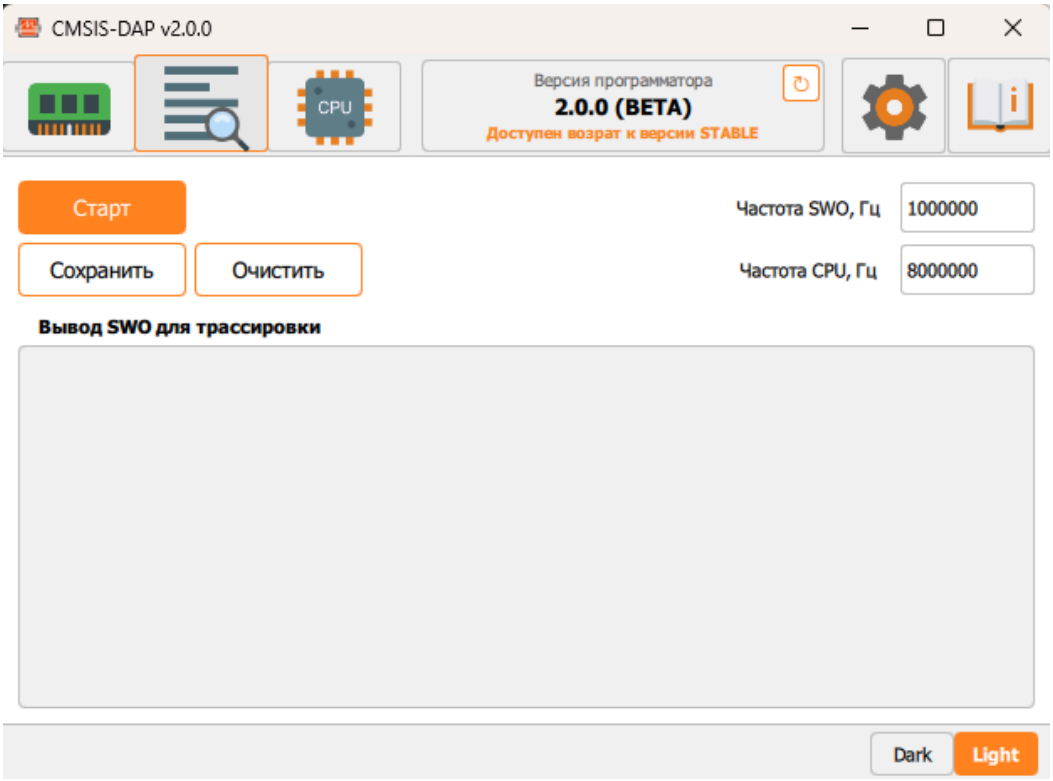

Рисунок 8 — Вкладка отображения вывода данных SWO

Для получения данных необходимо:

1. задать желаемую частоту обмена по интерфейсу SWO (не более 1000000), и указать текущую частоту работы микросхемы;
2. нажать на кнопку «Старт».

Для сохранения полученных данных в текстовый файл необходимо нажать кнопку «Сохранить». Кнопка «Очистить» очищает поле вывода данных SWO.

> **Примечание**
>
> SWO недоступен, если выбран интерфейс подключения JTAG или выполняются операции работы с памятью.

2.3. Вкладка доступа к регистрам ядра и переменным в памяти

Третья вкладка – «Доступ к регистрам ядра и переменным в памяти» (рис. 9), в которой осуществляется чтение и запись регистров ядра, переменных в памяти. Для этого нужно нажать кнопку «Подключить», после чего станут активными поля ввода данных. При работе с памятью можно выбрать в каком виде отображать значение переменной в памяти (двоичное, шестнадцатеричное или десятичное).

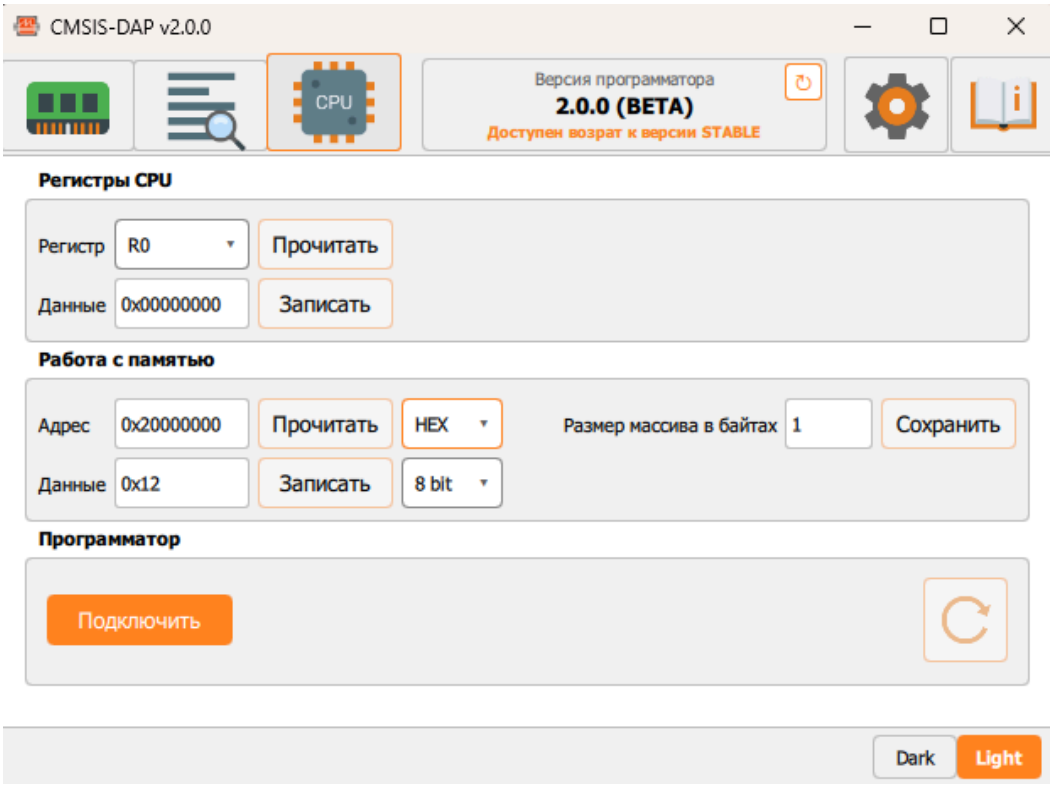

Рисунок 9 — Вкладка доступа к регистрам ядра и переменным в памяти

При чтении регистров ядра происходит его кратковременная остановка на время порядка 5 мс. Нужно учитывать это, если программа выполняет управление оборудованием, критичным к времени отклика.

При нажатии на кнопку сброса выполняется формирование аппаратного сигнала RESET (вывод RESET переводится в низкий уровень на ~100 мс).

2.4. Настройки программы

При нажатии на кнопку настроек появится окно с настройками программы (рис. 10).

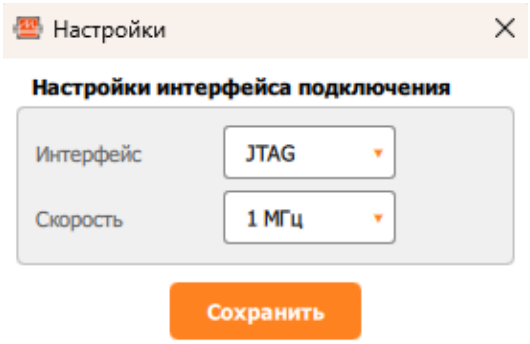

Рисунок 10 — Окно "Настройки"

Поддерживаются интерфейсы подключения JTAG и SWD на скоростях 100 кГц, 500 кГц и 1 МГц. Для применения изменений необходимо нажать кнопку «Сохранить».

2.5. Обновление ПО программатора

Начиная с версии 2.0.0 в интерфейс программы добавлена возможность обновления ПО программатора без использования сторонних средств. В верхней части окна программы расположена кнопка, которая в обычном состоянии отображает статус подключения программатора и текущую версию его ПО (рис. 11).

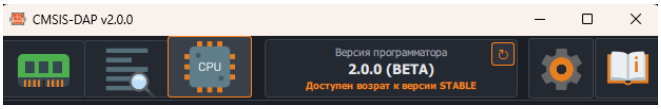

Рисунок 11 — Кнопка обновления ПО программатора

При нажатии на кнопку доступен выбор версии ПО программатора – **STABLE** (1.0.3) или **BETA** (2.0.0), см. Раздел 1. Во время обновления на программаторе будет активен синий светодиод (табл. 2), а в нижней части окна программы отображается ползунок хода выполнения.

> **Внимание**
>
> Не отключайте программатор от ПК во время обновления.

По завершении процедуры отображается сообщение об успешном завершении обновления. При работе с версией 2.0.0 (BETA) доступен возврат к версии 1.0.3 (STABLE).

> **Примечание**
>
> При возникновении замечаний по работе с версией 2.0.0 (BETA) просьба направлять сообщения в отдел технической поддержки по адресу support@milandr.ru.

3. Назначение выводов разъема программатора

Назначение выводов разъема программатора в различных режимах работы приведено на рис. 12.

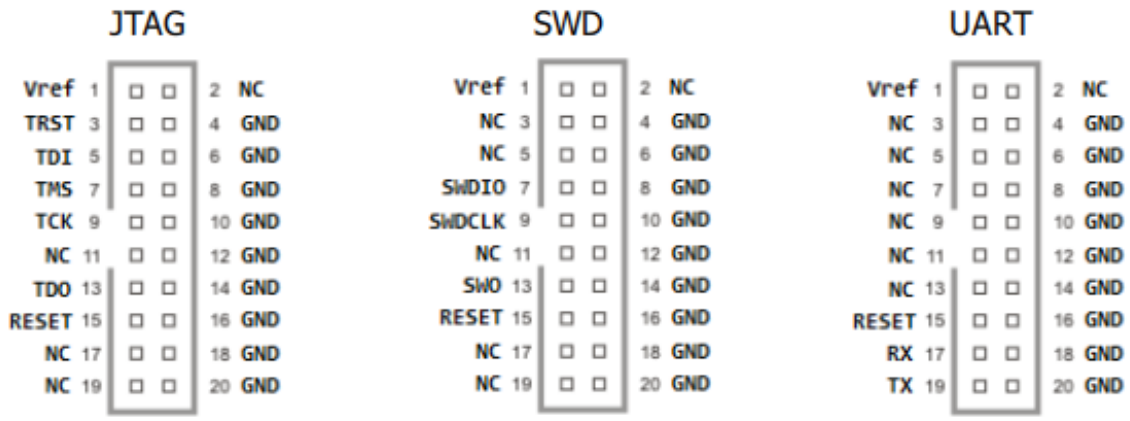

Рисунок 12 — Назначение выводов разъема программатора

Для работы программатора необходимо наличие напряжения на выводе Vref. Допустимый диапазон напряжений (2,7 - 5,5) В.

> **Внимание**
>
> Подать питание на целевую плату с данного программатора невозможно!

4. Описание световой индикации программатора

На верхней части корпуса расположен полноцветный светодиод. В зависимости от текущего состояния он может светиться разными цветами (см. табл. 2).

Таблица 2 — Описание световой индикации программатора

Состояние светодиода	**Описание**
Мигает красный	Нет подключения по USB
Горит красный	Программатор в режиме ожидания
Горит синий	Программатор в режиме обновления прошивки
Однократно моргнул синий	Вывод RESET был переведён в низкий, а потом в высокий уровень
Горит зелёный	Программатор подключен к микроконтроллеру
Моргает зелёный	Программатор в режиме UART передаёт данные
Моргает красный	Программатор в режиме UART принимает данные

Если программатор используется в связке со средой разработки, то состояние светодиода определяется этой средой. Например, в среде разработки IAR постоянно горящий желтый цвет сигнализирует о том, что программа сейчас выполняется, а зелёный – программа приостановлена.

5. Описание функционала программы

5.1. Работа с памятью

Для записи данных в FLASH-память микроконтроллера программа использует механизм загрузчиков. Загрузчик представляет собой скомпилированную под целевой микроконтроллер программу с функциями стирания, записи и проверки данных. Данная программа загружается в ОЗУ микроконтроллера.

Программа совместима с загрузчиками, используемыми в IDE IAR. Для добавления своего загрузчика необходимо создать загрузчик в соответствии с документом http://supp.iar.com/filespublic/updinfo/004916/arm/doc/flashloaderguide.pdf.

Далее загрузчик компилируется, и в папку «flashloaders_ceramic» или «flashloaders_plastic», расположенную в директории программы CMSIS-DAP.exe, добавляются 3 файла:

1. loader.flash
2. loader.out
3. script.js

***.flash**

В данном файле описывается FLASH-память. Теги macro и aggregate в программе не используются. Добавлены дополнительные теги:

- uart – если стоит 1, то есть UART-загрузчик в данном микроконтроллере. Поддерживается только UART-загрузчик фирмы АО «ПКК Миландр»;
- uart_baud – скорость UART, на которой производится синхронизация с загрузчиком;
- js – файл с JS-кодом.

***.out**

ELF-файл с программой загрузчика FLASH-памяти.

***.js**

Опциональный файл с JS-кодом. Выполняется перед чтением, записью и стиранием памяти. В нём можно отключить специфичную защиту, остановить Watchdog и выполнить прочие подготовительные действия.

При запуске программы происходит сканирование директорий «flashloaders_ceramic» и «flashloaders_plastic» на наличие файлов *.flash. При нахождении данного файла происходит попытка открытия файла *.out, указанного в теге exe. Полный

путь файла игнорируется, используется только его имя. При успешном открытии файла в список загрузчиков добавляется пункт с именем, как у файла *.flash.

Поддерживается возможность полного стирания при инициализации памяти. Для этого выставляется флаг FLAG_ERASE_ONLY при выполнении FlashInitEntry. Если загрузчик не поддерживает такую возможность, то будет выполнено стирание всей памяти по секторам.

Для проверки CRC записанных данных вызывается функция FlashChecksumEntry. Если данная функция не реализована в загрузчике, то происходит чтение записанных данных с последующим вычислением CRC.

5.2. Встроенные загрузчики

5.2.1. Микроконтроллеры в металло-керамическом корпусе

5.2.1.1. K1986VE1x_002TU

Загрузчик для К1986ВЕ1Т (12134), К1986ВЕ1АТ (12144). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB. Сектор А (1 КБ) содержит производственную информацию: ID и калибровочные значения.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без сектора А (3 КБ). Стирание и программирование сектора А информационной области игнорируются.
- `"--info_memory_factory"` - работа со всей информационной областью (4 КБ).
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область без сектора А (3 КБ).
- `"--erase_all --info_memory_factory"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.1.2. K1986VE9x_002TU

Загрузчик для К1986ВЕ91Т (12104), К1986ВЕ92У(1) (12115(А)), К1986ВЕ93У (12125), К1986ВЕ94Т (12094). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.

- Информационная область (4 КБ): данные, доступные через APB. Сектор A (1 КБ) содержит производственную информацию: ID и калибровочные значения.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без сектора A (3 КБ). Стирание и программирование сектора A информационной области игнорируются.
- `"--info_memory_factory"` - работа со всей информационной областью (4 КБ).
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область без сектора A (3 КБ).
- `"--erase_all --info_memory_factory"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.1.3. K1901VC1T

Загрузчик для К1901ВЦ1Т (К1901ВЦ1Т). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.1.4. K1986VE1x

Загрузчик для К1986ВЕ1Т (К1986ВЕ1Т), К1986ВЕ1АТ (К1986ВЕ1АТ). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)

- информационная область (4 КБ).

5.2.1.5. K1986VE3T

Загрузчик для К1986ВЕ3Т (К1986ВЕ3Т). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.1.6. K1986VE4U

Загрузчик для К1986ВЕ4У (К1986ВЕ4У). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (8 КБ): загрузчик и данные, доступные через AHB и APB.

Производственный загрузчик расположен в блоке 0 (2 КБ) информационной области. Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без блока 0 (6 КБ). Программирование и стирание блока 0 (загрузчика) игнорируются.
- `"--info_memory_boot"` - работа со всей информационной областью (8 КБ).
- `"--restore_boot"` - восстановление производственного загрузчика в блоке 0 при выполнении одиночной операции стирания.
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область без блока 0 (6 КБ).
- `"--erase_all --info_memory_boot"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (8 КБ).

Основная область, 4 блока * 32 КБ: 0x0000_0000-0x0001_FFFF.

Информационная область, 4 блока * 2 КБ (адресация с разрывами):

- блок 0: 0x0000_0000-0x0000_0800
- блок 1: 0x0000_8000-0x0000_8800
- блок 2: 0x0001_0000-0x0001_0800
- блок 3: 0x0001_8000-0x0001_8800

5.2.1.7. K1986VE9x

Загрузчик для К1986ВЕ91Т (К1986ВЕ91Т), К1986ВЕ92У(1) (К1986ВЕ92У(1)), К1986ВЕ93У (К1986ВЕ93У), К1986ВЕ94Т/Ф/Я (К1986ВЕ94Т/Ф/Я). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.1.8. K1986VK01x

Загрузчик для К1986ВК018 (К1986ВК018), К1986ВК016 (К1986ВК016). Дополнительные опции не реализованы.

5.2.2. Микроконтроллеры в пластмассовом корпусе

5.2.2.1. K1986VE1xI

Загрузчик для К1986ВЕ1FI (MDR1213FI), К1986ВЕ1GI (MDR1213GI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные с доступом через AHB и APB.
- Информационная область (4 КБ): данные с доступом через APB. Сектор А (1 КБ) содержит производственную информацию: ID и калибровочные значения.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без сектора А (3 КБ). Стирание и программирование сектора А информационной области игнорируются.
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область без сектора А (3 КБ).

5.2.2.2. K1986VE9xI

Загрузчик для К1986ВЕ92F(1)I (MDR1211F(1)I), К1986ВЕ94GI (MDR1209GI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные с доступом через AHB и APB.
- Информационная область (4 КБ): данные с доступом через APB. Сектор А (1 КБ) содержит производственную информацию: ID и калибровочные значения.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (3 КБ без сектора А). Стирание и программирование сектора А информационной области игнорируются.
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (3 КБ без сектора А).

5.2.2.3. K1901VC1QI

Загрузчик для К1901ВЦ1QI (MDR32FG16S1QI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.2.4. K1986VE1QI

Загрузчик для К1986ВЕ1QI (MDR32F1QI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание затрагивает две области:
  - основная область (128 КБ)

- информационная область (4 КБ).

5.2.2.5. K1986VE92QI

Загрузчик для К1986ВЕ92QI (MDR32F9Q2I). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): данные, доступные через APB.

Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью (4 КБ).
- `"--erase_all"` - полное стирание затрагивает две области:
  - основная область (128 КБ)
  - информационная область (4 КБ).

5.2.2.6. K1986VK01GI

Загрузчик для К1986ВК01GI (MDR1201GI). Дополнительные опции не реализованы.

5.2.2.7. K1986VK214

Загрузчик для К1986ВК214 (MDR32F23QI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (64 КБ): код и данные, доступные через AHB и APB.
- Информационная область (4 КБ): загрузчик и данные, доступные через AHB и APB.

Производственный загрузчик расположен в блоке 0 (2 КБ) информационной области. Без опций загрузчик работает с основной областью (64 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без блока 0 (2 КБ). Программирование и стирание блока 0 (загрузчика) игнорируются.
- `"--info_memory_boot"` - работа со всей информационной областью (4 КБ).
- `"--restore_boot"` - восстановление производственного загрузчика в блоке 0 при выполнении одиночной операции стирания.
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (64 КБ)
  - информационная область без блока 0 (2 КБ).
- `"--erase_all --info_memory_boot"` - полное стирание, затрагивает две области:
  - основная область (64 КБ)
  - информационная область (4 КБ).

Основная область, 2 блока * 32 КБ: 0x0000_0000-0x0000_FFFF.

Информационная область, 2 блока * 2 КБ (адресация с разрывами):

- блок 0: 0x0000_0000-0x0000_0800
- блок 1: 0x0000_8000-0x0000_8800

5.2.2.8. K1986VK234

Загрузчик для К1986ВК234 (MDR32F21QI). Внутренняя FLASH-память микроконтроллера состоит из двух областей:

- Основная область (128 КБ): код и данные, доступные через AHB и APB.
- Информационная область (8 КБ): загрузчик и данные, доступные через AHB и APB.

Производственный загрузчик расположен в блоке 0 (2 КБ) информационной области. Без опций загрузчик работает с основной областью (128 КБ).

Опции:

- `"--info_memory"` - работа с информационной областью без блока 0 (6 КБ). Программирование и стирание блока 0 (загрузчика) игнорируются.
- `"--info_memory_boot"` - работа со всей информационной областью (8 КБ).
- `"--restore_boot"` - восстановление производственного загрузчика в блоке 0 при выполнении одиночной операции стирания.
- `"--erase_all"` - полное стирание с защитой, затрагивает две области:
  - основная область (128 КБ)
  - информационная область без блока 0 (6 КБ).
- `"--erase_all --info_memory_boot"` - полное стирание, затрагивает две области:
  - основная область (128 КБ)
  - информационная область (8 КБ).

Основная область, 4 блока * 32 КБ: 0x0000_0000-0x0001_FFFF.

Информационная область, 4 блока * 2 КБ (адресация с разрывами):

- блок 0: 0x0000_0000-0x0000_0800
- блок 1: 0x0000_8000-0x0000_8800
- блок 2: 0x0001_0000-0x0001_0800
- блок 3: 0x0001_8000-0x0001_8800

6. Алгоритм вычисления CRC

При открытии файла с данными, сверке и считывании данных отображается CRC данных. Алгоритм вычисления CRC приведен в лист. 1.

Листинг 1 — Алгоритм вычисления CRC

```
uint16_t calcCrc16( uint8_t *data, uint32_t size, uint16_t crc )
{
    while (size--)
    {
        int i;
        uint8_t byte = *data++;

        for (i = 0; i < 8; ++i)
        {
            uint32_t osum = crc;
            crc <<= 1;
            if (byte & 0x80)
                crc |= 1;
            if (osum & 0x8000)
                crc ^= 0x1021;
            byte <<= 1;
        }
    }
    return crc;
}
```

Для совместимости с функцией Crc16 из загрузчика, CRC данных считается в соответствии с лист. 2.

Листинг 2 — Расчет CRC для данных

```
uint8_t zero[2] = { 0, 0 };
uint16_t crc = 0;

crc = _calcCrc16( someData, someDataSize, crc );
crc = _calcCrc16( zero, 2, crc );
```

7. Работа со скриптами

В программе реализована возможность выполнять JS-код из файла перед записью, стиранием и чтением памяти. Для этого необходимо в файл .flash добавить тег, как показано в лист. 3.

Листинг 3 — Подключение скрипта в .flash-файле

```
<flash_device>
  ...
  <js>script.js</js>
  ...
</flash_device>
```

Для возможности определения текущего события добавлена глобальная переменная event. Пример с её возможными состояниями приведен в лист. 4.

Листинг 4 — Определение состояния event

```
if( event == "save" )
{
    // скрипт вызван перед процедурой чтения памяти в файл
}
else if( event == "program" )
{
    // скрипт вызван перед процедурой записи в память
}
else if( event == "erase" )
{
    // скрипт вызван перед процедурой стирания всей памяти
}
```

На текущий момент доступны следующие функции:

- `void dap.showMessage("Text")` – отображение сообщения в статусной строке программы;
- `unsigned int dap.readMemory32( unsigned int adr )` – чтение памяти по заданному адресу;
- `bool dap.writeMemory32( unsigned int adr, unsigned int val )` – запись памяти по заданному адресу. Возвращает true при успешной записи;
- `unsigned int dap.readDpReg( unsigned int reg )` – чтение регистра Debug port;
- `bool dap.writeDpReg( unsigned int reg, unsigned int val )` – запись регистра Debug port. Возвращает true при успешной записи;
- `unsigned int dap.readApReg( unsigned int reg )` – чтение регистра Access port;
- `bool dap.writeApReg( unsigned int reg, unsigned int val )` – запись регистра Access port. Возвращает true при успешной записи.

Пример скрипта для микроконтроллера с ядром Cortex-M3 приведен в лист. 5.

Листинг 5 — Пример скрипта для микроконтроллера с ядром Cortex-M3

```
var idcode, cpuid, dhcsr, result;

idcode = dap.readDpReg( 0x00 ); // Чтение IDCODE (0x00 - DP_IDCODE)
dap.showMessage("IDCODE = 0x" + idcode.toString(16).toUpperCase());

cpuid = dap.readMemory32( 0xE000ED00 ); // Чтение CPUID (0xE000ED00)
dap.showMessage("CPUID = 0x" + cpuid.toString(16).toUpperCase());

dhcsr = dap.readMemory32( 0xE000EDF0 ); // Чтение DHCSR (0xE000EDF0)
if( (dhcsr & 0x20000) === 0 ) // Проверка бита S_HALT
{
    dap.showMessage("Core is running");
}
else
{
    dap.showMessage("Core is halted");
}

result = "Success";
```

8. Решение проблем

В случае возникновения сообщения об ошибке или некорректного функционирования программы можно создать диагностический файл. Для этого необходимо запустить программу с ключом -d. В результате в директории программы будет создан файл logFile.txt.

Если Вы обнаружите ошибку в этом документе или проблему в программном обеспечении CMSIS-DAP, пожалуйста, сообщите нам об этом (support@milandr.ru), и мы постараемся помочь Вам как можно скорее.

9. Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	Описание
1.	—	–	Не фиксировались	-
2.	06.11.2020	2.5.1	Добавлены требования к используемому hex-файлу	По тексту
3.	14.05.2021	2.5.2	Исправлены опечатки	Стр.12
4.	14.07.2022	2.6.0	Заменены рисунки, добавлены ссылки на рисунки и таблицы, добавлено описание загрузчиков	
5.	07.08.2026	2.7.0	Добавлено описание протокола CMSIS-DAP v2, актуализирован раздел обновления ПО программатора, добавлено описание тёмной темы оформления интерфейса, добавлен раздел системных требований, актуализирован раздел настроек программы, добавлено описание переключателя типа корпуса микроконтроллера при выборе загрузчика	По тексту
6.	04.09.2026	2.7.1	Уточнены требования к используемому hex-файлу (неактуальны для версий ПО 2.0.0 и выше); уточнен текст об остановке ядра при чтении регистров ядра	Стр. 2, 11