Общее описание

***Jetlink*Флеш** – устройство для программирования ARM микроконтроллеров с внутренней или внешней Flash памятью. Может использоваться как в автономном режиме (без подключения к компьютеру) так и с программой JFlash. Дополнительно JetlinkФлеш имеет все возможности обычного эмулятора Jetlink.

***Jetlink*Флеш** подключается к компьютеру работающему под операционной системой Windows 2000, Windows XP, Windows 2003, Windows Vista или Windows7 через USB или интерфейс RS-232. Эмулятор имеет стандартный 20-контактный разъем JTAG.

Возможности

- 3 режима работы: режим эмулятора Jetlink, Автономный режим и MSD режим
- Не требует внешнего блока питания, работает от USB
- Поддерживает ARM7/ARM9/Cortex-M3 микроконтроллеры
- Поддерживает программирование внутренней и внешней памяти
- 128 МБ для хранения файлов прошивки
- Поддержка SWD

Режимы работы

Эмулятор может работать в 3 режимах:

- Режим обычного эмулятора Jetlink
- Автономный режим
- MSD режим (Mass Storage Device)

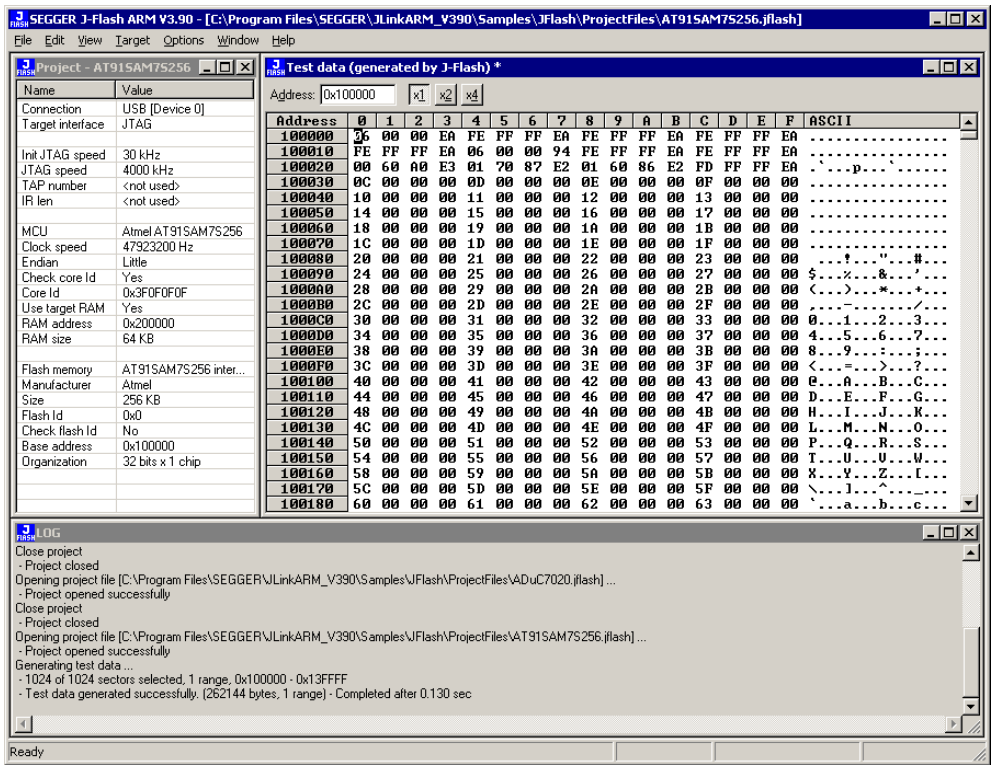
При подключении к компьютеру через USB эмулятор работает в режиме Jetlink, если питание на USB подано (внешний блок питания), но енумерация не прошла, эмулятор переключается в Автономный режим, если нажать кнопку Старт/Стоп и подключить к компьютеру по USB эмулятор включится в режим MSD.

Порядок подключения

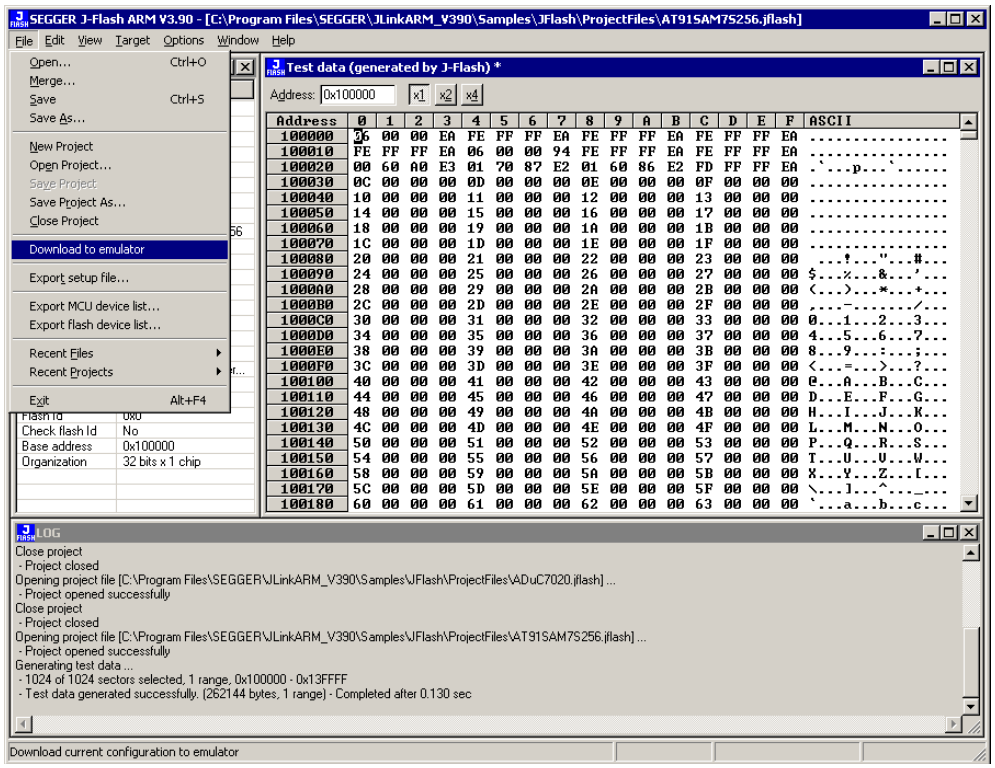
Мы рекомендуем сначала подключить эмулятор (USB или отдельный блок питания), а затем подключать к отлаживаемой/программируемой плате.

Подготовка к использованию в Автономном режиме

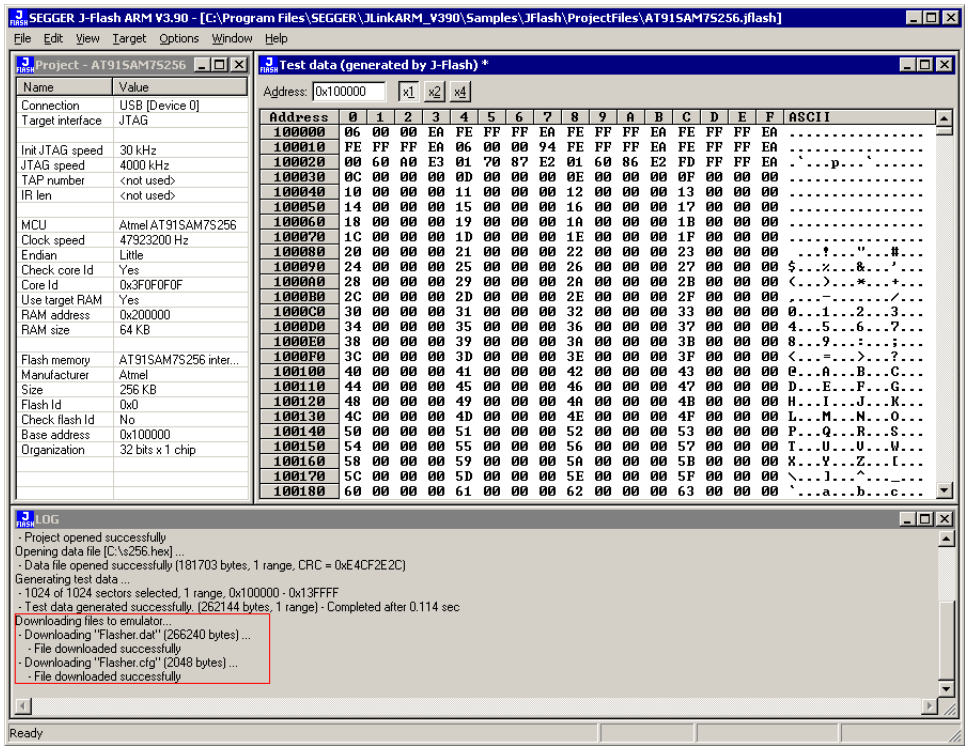
Перед использованием эмулятора в Автономном режиме, вам необходимо записать в эмулятор файлы проекта и файл прошивки. Для этого подключите эмулятор в обычном Jetlink режиме и запустите программу J-Flash.exe



Откройте в ней нужный файл проекта (File-Open Project) и файл прошивки (File- Open Data File). В настройках проекта произведите нужные изменения, а затем загрузите данную конфигурацию в эмулятор (File-Download to programmer).



После загрузки, в логе программы вы должны увидеть сообщение о удачной загрузке файлов.



Теперь эмулятор готов к работе в Автономном режиме. Подключите эмулятор к к внешнему блоку питания кабелем USB и к программируемой плате. Нажатие кнопки Start/Stop будем приводить к запуску процесса программирования выбранного вами файла с настройками указанными в проекте.

MSD режим

Этот режим позволяет записать файлы в эмулятор не используя программу J-Flash. Нажмите кнопку Start/Stop и подключите эмулятор к компьютеру. Windows обнаружит новое съемное устройство хранения размером 128 МБ.
Вы должны записать следующие файлы –
Flasher.cfg – содержит конфигурацию настроек для программируемого устройства.
Flasher.dat – содержит данные для программирования
Flasher.log – содержит всю информацию о выполненных командах в Автономном режиме
Serial.txt – содержит серийный номер, который будет запрограммирован следующим.
Два последних файла не являются обязательными.

Поддержка множества файлов.

Эмулятор позволяет хранить в памяти несколько файлов настроек и данных.
Для выбора необходимой конфигурации используется файл Flasher.ini, в котором должна быть следующая запись

```
[FILES]
DataFile = “Flasher1.dat”
```

ConfigFile = “Flasher1.cfg”

Используя данный метод можно загрузить множество файлов конфигурации и данных во внутреннюю память устройства и выбирать нужную пару изменениями Flasher.ini, двумя способами:

- Загрузить эмулятор в MSD режим и заменить Flasher.ini
- Используя интерфейс RS-232, с помощью IO команд.

Интерфейс программирования

Поддерживаются два интерфейса:

- JTAG
- SWD

Поддержка внешних микросхем Flash памяти

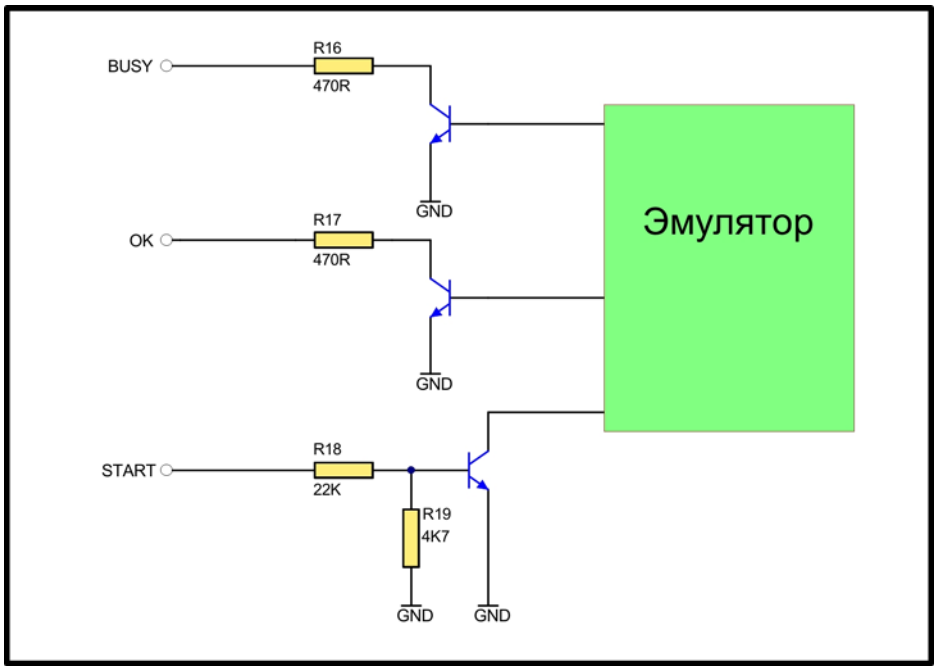
Поддерживается CFI – совместимое программирование внешней NOR памяти.
Список поддерживаемых микросхем

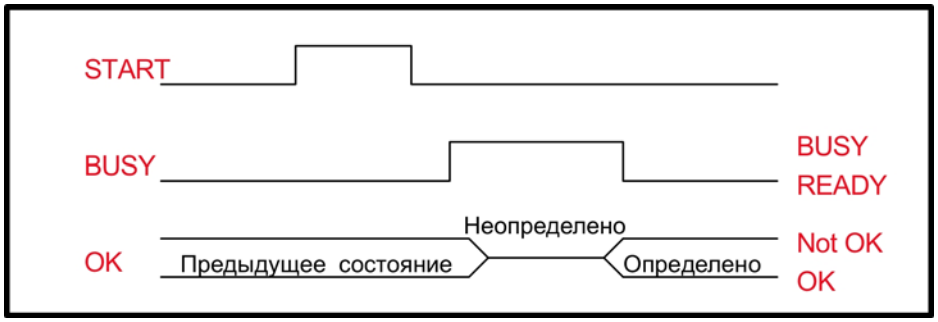
Удаленное управление

Имеется три способа для управления эмулятором в автономном режиме:

- Ручной: Программирование осуществляется нажатием кнопки Start/Stop
- Используя 3 линии управления. Первая линия для старта операции программирования, вторая для контроля Busy (занятости устройства) и третья для статуса.
- Управление по RS-232 интерфейсу.

Все три способа работают только в Автономном режиме. В Jetlink и MSD режиме они не используются.





Вывод разъема	Функция	Описание
1	START	Положительный импульс амплитудой 5-30В с длительностью минимум 30мс запускает Auto режим (Стирание/Программирование/Верифицирование) по спадающему фронту импульса.
4	BUSY	Как только режим Auto запущен, сигнал Busy становится активным
5	GND	
7	OK	Вывод результата работы.

ASCII управляющий интерфейс

Формат сообщений

- Любая ASCII команда начинается символом #.
- Любая ASCII команда должна заканчиваться «Возврат каретки» (ASCII код 13)
- Команды могут посылаться в верхнем или нижнем регистре.

Настройки RS-232

Эмулятор управляется через последовательный интерфейс RS-232 с следующими настройками:

- 8 бит данных
- Нет четности
- 1 стоп бит
- Скорость 9600 бод

Команды

#AUTO – соответствует нажатию кнопки на эмуляторе. Обычно данная команда выполняет следующую последовательность

- Эмулятор запускает стирание
- Эмулятор запускает программирование
- Эмулятор верифицирует запрограммированное

В зависимости от настроек проекта (FLASHER.cfg) последовательность может быть иной. По окончанию операции эмулятор отвечает
#OK – если ошибок не обнаружено
#ERRxxx если произошла ошибка.

В процессе выполнения команды AUTO эмулятор выдает статусные сообщения. Например –

```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#STATUS:PROGRAMMING
#STATUS:VERIFYING
#OK (Total 13.993s, Erase 0.483s, Prog 9.183s, Verify 2.514s)
```

#AUTO NOINFO – выполняет тоже самое, что и AUTO, но не выдает статусные сообщения в процессе работы. Данная команда возвращает **#OK** или **#ERRxxx**

#ERASE – данная команда выполняет функцию Стереть Все (Erase All) выбранные сектора.

```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#STATUS:UNLOCKING
#STATUS:ERASING
#OK (Total 0.893s, Erase 0.483s)
```

#START – данная команда используется для старта прошивки записанной с помощью программатора. Все сигналы JTAG/SWD переводятся в Z – состояние и производится сброс устройства.

```
#ACK
#STATUS:INITIALIZING
#STATUS:CONNECTING
#OK (Total 1.148s)
```

#STATUS – данная команда может быть выполнена в любое время выполнения другой команды. Эмулятор верней текущий статус. Все статусы описаны в таблице

#PROGRAM - эта команда используется аналогично команде AUTO, но не выполняет стирание и верификацию.

#VERIFY – команда используется для верификации записанных данных с содержимым программируемого файла.

#RESULT – эта команда может быть выполнена во время исполнения любой другой команды. Эмулятор вернет последний результат предыдущей выполненной команды.

#CANCEL – данная команда может быть использована для прерывания программирования. Эмулятор вернет

```
#ERR007:CANCELED.
```

#BAUDRATE<Baudrate> – эта команда используется для изменения скорости RS-232. **<Baudrate>** в десятичном формате. На эту команду эмулятор вернет

```
#ACK
#OK
```

Или одну из возможных ошибок –

```
#ERR255: Неизвестный параметр
или
#ERR255: Скорость не поддерживается
```

Замечание: После посылки команды **#BAUDRATE** вы должны дождаться пока эмулятор ответит #OK. Рекомендуется выждать 5 мс перед посылкой следующей команды на новой скорости, дав эмулятору время произвести перенастройку.

Команды FILE I/O

ASCII интерфейс поддерживает следующие I/O команды:

#FOPEN<FileName> команда открывает файл FileName для дальнейших операций. Если параметр **<FileName>** не указан – эмулятор создаст новый файл.
Типовая последовательность –

```
#FOPEN flasher.dat
#ACK
#OK
```

Замечание: В настоящее время поддерживается работа только с одним файлом в одно и тоже время. Если **#FOPEN** команда будет послана, когда открыт другой файл эмулятор вернет:

```
#ACK
#ERR255:A file has already been opened
```

#FCLOSE – команда закрывает открытый файл в эмуляторе, который был открыт командой **#FOPEN**. Типовая последовательность –

```
#FCLOSE
#ACK
#OK
```

Замечание: Используйте команду **#FCLOSE** к файлу который был открыт командой **#FOPEN**. В противном случае эмулятор вернет –

```
#ACK
#ERR255:No file opened
```

#FDELETE<FileName> - команда удаляет файл в эмуляторе с именем **<FileName>**
Типовая последовательность –

```
#FDELETE flasher.dat
#ACK
#OK
```

Замечание: Если удаление не удалось или файл не найден, эмулятор вернет:

```
#ACK
#ERR255:Failed to delete file
```

#FWRITE<Offset>,<NumBytes>:<Data>
Команда служит для записи в файл, открытый командой #**FOPEN**. Параметр **<Offset>** задает смещение в файле от которого данные должны быть записаны. Параметр **<NumBytes**> задает количество байт, которые будут отправлены с этой командой для записи в файл. Параметр **<NumBytes>** ограничен максимальным значением 512. Таким образом, если вы хотите записать 1024 байта, вам необходимо послать команду **#FWRITE** дважды. **<Offset>** и **<NumBytes>** задаются в шестнадцатиричном формате.

```
#FWRITE 0,200:<Data>
#FWRITE 200,200:<Data>
```

Данные также должны быть представлены в шестнадцатиричном формате. Пример:

```
Data to be send: Hello !
ASCII values: 0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x20, 0x21
#FWRITE 0,7:48656C6C6F2021
```

#FREAD < Offset>,<NumBytes> - команда используется для чтения из файла, хранящегося в эмуляторе. **<Offset>** задает смещение в файле для чтения, а **<NumBytes>** задает число байт для чтения. Типовая последовательность использования команды

```
#FREAD 0,4
#ACK
#OK:04:466c6173
```

Если команда **#FREAD** успешно выполнена, эмулятор выдает сообщение

```
#OK:<Num-Bytes>:<Data>
```

Замечание: Перед использованием команды, файл должен быть открыт командой **#FOPEN**. В противном случае эмулятор выдаст сообщение:

```
#ACK
#ERR255:No file opened
```

#FSIZE – команда служит для получения информации о размере файле, открытого командой **#FOPEN**. Данные выдаются в 16-ричном формате. Пример:

```
#FSIZE
#ACK
#OK:10 // файл открытый в текущий момент имеет размер 16 байт
```

Ответы от эмулятора

#ACK – эмулятор выдает данное сообщение при приеме любой известной команды.
#NACK – эмулятор выдает при приеме неизвестной команды.

#OK – эмулятор выдает, если операция завершилась без ошибок.

#OK:<NumBytes>:<Data> - эмулятор выдает если команда **#FREAD** завершена. **<NumBytes>** - число прочитанных байт. Это значение может отличаться от количества запрошенных байт для чтения, например, если больше байт чем доступно было запрошено. **<Data>** - данные. **<NumBytes>** и **<Data>** имеют 16-ричный формат.

#OK:<Size> - эмулятор выдает, если команда **#FSIZE** завершена без ошибок. **<Size>** - размер файла в байтах в шестнадцатиричном формате.

#STATUS: - эмулятор сообщает текущее состояние.

Сообщение	Описание
#STATUS : READY	Эмулятор готов к приему следующей команды
#STATUS : CONNECTING	Эмулятор инициализирует подключение к программируемой микросхеме
#STATUS : INITIALIZING	Эмулятор производит самотестирование
#STATUS : UNLOCKING	Разблокировка секторов
#STATUS : ERASING	Стирание памяти в программируемой микросхеме
#STATUS : PROGRAMMING	Программирование
#STATUS : VERIFYING	Верифицирование памяти

#ERRxxx - Если любая команда, кроме **#STATUS #RESULT** завершится с ошибкой, эмулятор отменяет команду и отвечает сообщением об ошибке вместо **#OK** сообщения.
Некоторые коды ошибок могут иметь двоеточие и дополнительный текст ошибки.
Например:

```
#ERR007:CANCELED.
```

Сообщение	Описание
#ERR007	Эмулятор получил команду

	#CANCEL и отменил текущую операцию
#ERR255	Произошла неизвестная ошибка. За ней следует текст ошибки

JTAG/SWD разъем

Назначение контактов разъема

Вывод	**JTAG**	**SWD**	**Описание**
1	VREF	Вход	Напряжение питания программируемой схемы.
2	VSupply	NC	Не используется
3	nTRST	NC	Сброс JTAG порта
5	TDI	NC	Выход данных JTAG
7	TMS	SWDIO	Выход режима JTAG (I/O в режиме SWD)
9	TCK	SWCLK	Тактирование JTAG (I/O клок в SWD)
11	RTCK		Возвратный клок с JTAG
13	TDO	SWO	Вход данных JTAG (SWO в SWD)
15	RESET	I/O	Сигнала сброса
17	DBGRQ		Не используется
19	5V		Может быть использован для программируемой схемы.
4,6,8	GND		
10,12,14	GND		
16,18,20	GND		

RS-232 разъем