Руководство по загрузке и отладке программы в микроконтроллере 1923ВК014

Микросхема 1923ВК014 – специализированный микроконтроллер внешней памяти и набора периферии на базе микропроцессорного ядра ARM Cortex-M0. В первой ревизии МК интерфейс для отладки JTAG не доступен, поэтому для загрузки и отладки программы необходимо воспользоваться UART загрузчиком.

Требования к оснастке

Режим запуска задается комбинацией сигналов на входе MODE[7:0]. При этом режим закодирован кодом Хемминга (7,4), таким образом, что биты MODE[3:0] задают режим, а биты MODE[7:4] задают проверочные ECC биты. Для того, чтобы выбрать режим UART загрузчика необходимо, согласно документации, на выводах MODE[7:0] ({PE[5:0], PD[31:30]}) задать один из следующих режимов:

Таблица 1. Режимы запуска UART загрузчика

Биты ECC MODE[7:4]	Биты режима MODE[3:0]	Режим	Краткое описание
0010	1011	UART0+JB	Загрузка последовательно из внешней памяти по UART0 интерфейсу с включенным интерфейсом отладки через выводы JTAG_B
0011	1100	UART0+JA	Загрузка последовательно из внешней памяти по UART0 интерфейсу с включенным интерфейсом отладки через выводы JTAG_A

При использовании режима UART0+JA на плате необходимо реализовать подключение к выводам UART0:

PE15 – TX
PE16 – RX

микросхемы приёмопередатчика по стандарту RS-232/485 и обеспечить наличие соответствующего разъёма для соединения с ПК с помощью переходника USB-RS-232/485.

При использовании режима UART0+JB приёмопередатчик необходимо подключать к другим выводам UART0:

PA7 – TX
PA8 – RX

Загрузка программы

При старте микроконтроллера запускается BootLoader, который проверяет выводы MODE и настраивает аппаратный UART загрузчик с параметрами согласно спецификации:

- скорость 9600 бод
- количество бит данных - 8
- четность - выкл
- 1 стоп бит

Для загрузки программы через UART интерфейс необходимо воспользоваться программой Loader, которая позволяет загружать пользовательские программы во внутреннюю память ОЗУ МК или внешнюю Flash 1636РР3У. В ходе работы программы в ОЗУ МК загружается либо пользовательская программа, либо загрузчик filefl.hex, реализующий загрузку пользовательской программы во внешнюю Flash память.

Утилита Loader не требует установки, но для её запуска нужны библиотеки Visual Basic Powerpacks 10, которые необходимо скачать и установить. (http://download.microsoft.com/download/A/D/E/ADEFBFFF-2165-4F63-BB29-DCE891B95CC7/VisualBasicPowerPacksSetup.exe)

Перед тем как приступать к работе с программой, необходимо провести инициализацию COM порта. Для этого сначала нужно соединить плату с МК посредствам переходника USB-RS-232/RS-485 с ПК, после чего подать на плату основное питание. На компьютере в диспетчере устройств необходимо узнать номер COM порта и задать соответствующее значение в программе Loader.

В поле программы *"BR"* указываем скорость обмена. Можно выбрать либо 9600, либо 115200. Так как первоначально UART в МК настроен на скорость 9600, то при выборе 115200 будет подана команда перехода на новую скорость. После выбора этих двух параметров нажимаем кнопку *"Init Port"*, в окошке консоли отобразится надпись *"Port opened!"*.

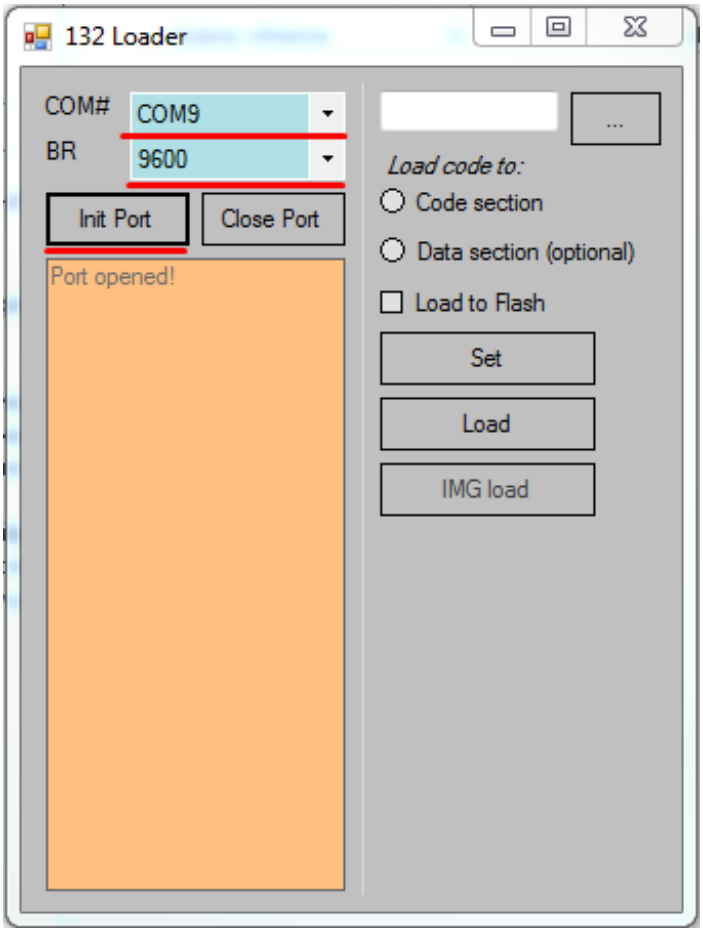


Рисунок 1. Инициализация COM-порта

После инициализации COM порта и успешного соединения можно загружать программу в МК. Программа для загрузки должна быть в формате *.hex.

При компиляции программы в среде Keil файл данного формата генерируется автоматически. Однако необходимо правильно указать настройки для загрузки программы в ОЗУ. Пример для Keil:

Размещение программы в области ОЗУ программ (Code Section)

Размещение программы в области ОЗУ данных (Data Section)

Чтобы загрузить программу в ОЗУ память МК в утилите Loader необходимо:

1. Нажать кнопку "..." и выбрать файл прошивки формата *.hex.

Указать область для который был сгенерирован *.hex файл.

- Code section;
- Data section (optional).

Галочка у опции «Load to Flash» должна быть снята.

2. Нажать клавишу Set.

3. Нажать клавишу Load.

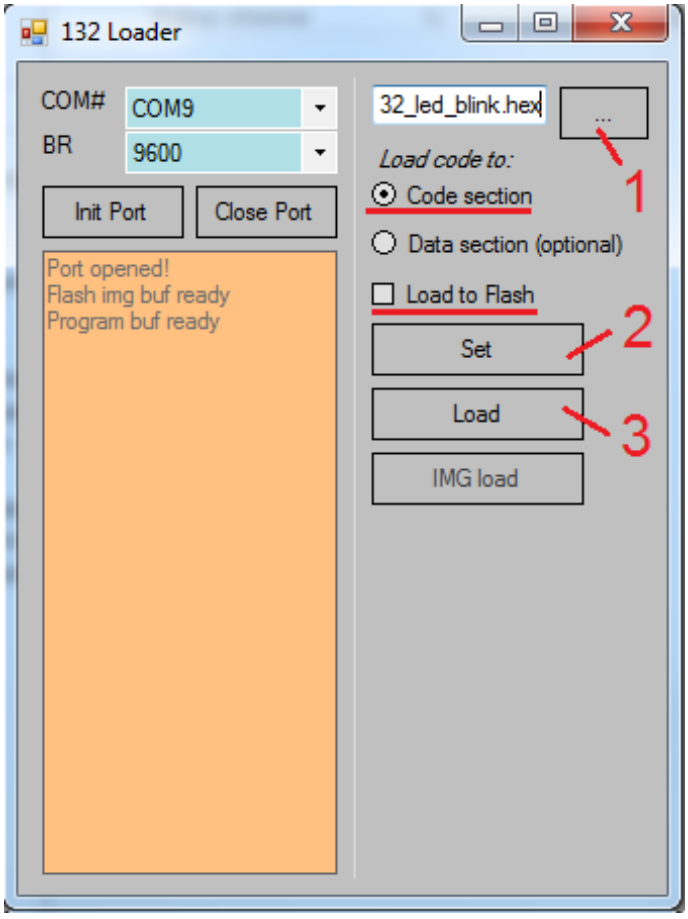


Рисунок 2. Механизм загрузки в область Code section

Если при загрузке программы в МК не удается синхронизироваться (Sync error), попробуйте сбросить МК кнопкой Reset и повторить процедуру записи.

Для загрузки программы во внешнюю Flash память 1636РР3У в утилите Loader необходимо:

1. Нажать кнопку "..." и выбрать файл прошивки формата *.hex. Указать область «Code section», *.hex файл также должен быть сгенерирован для данной области. Установить галочку у опции «Load to Flash».

2. Нажать клавишу Set.

3. Установить на плате перемычку SPI SEL и нажать клавишу Load, после чего будет произведена загрузка в ОЗУ МК загрузчика Flash filefl.hex, который должен находиться в одном каталоге с программой.

4. После появления в консоли программы сообщения «*Ready to load flash image!*» нажать клавишу IMG load. Если программа имеет размер более 1024 слов, то она разбивается и загружается по частям.

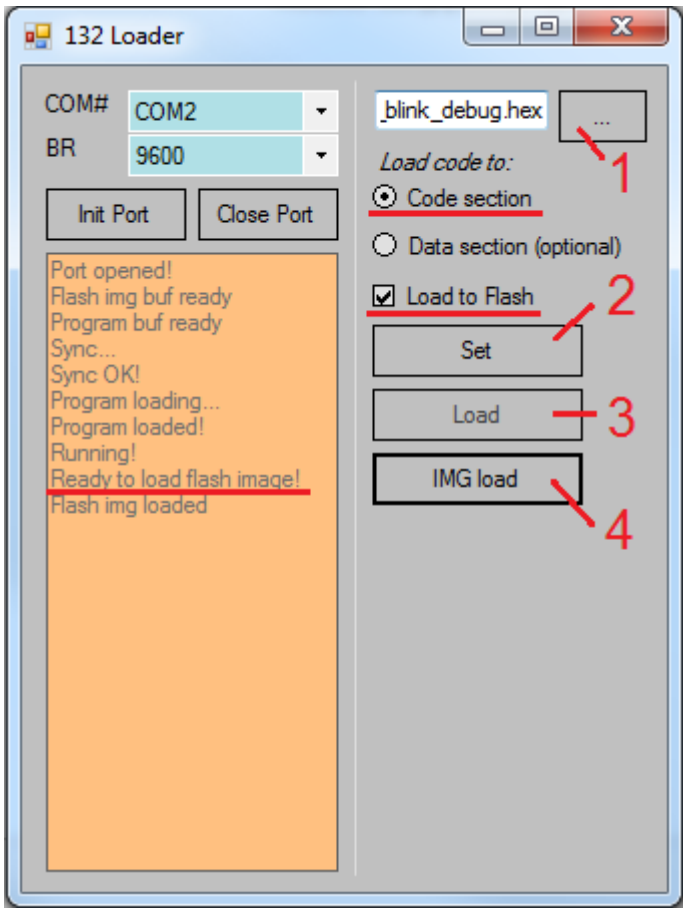


Рисунок 3. Механизм загрузки во Flash память

Для проверки работы программы необходимо выключить питание на плате, изменить режим загрузки МК на SPI2+JB (MODE[7:0]=10011001) или SPI3+JA (MODE[7:0]=01011010), после чего снова включить питание на плате.

Отладка программы

Отладку программы в МК 1923ВК014 можно осуществлять с использованием отладчика GDB. GDB осуществляет удалённую отладку двумя способами: с использованием на целевой машине (МК) либо отладочной заглушки (GDB stub), либо программы GDB сервер (gdbserver). Последний вариант с GDB сервером работает только в Unix-подобных системах, поэтому для МК 1923ВК014 будем использовать отладочную заглушку.

Отладочная заглушка представляет собой набор подпрограмм специального назначения, которые компилируются вместе с пользовательской программой, и реализуют удалённый последовательный протокол GDB.

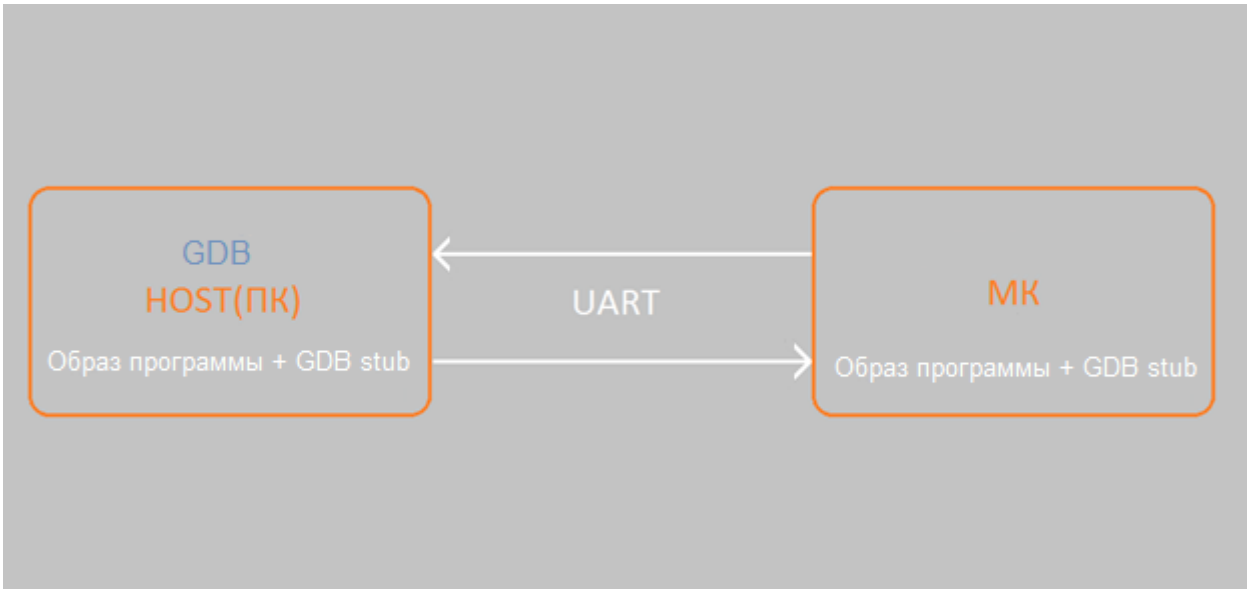


Рисунок 4. Схема взаимодействия GDB с МК

Отладчик GDB можно использовать из бесплатного набора инструментов GNU Embedded Toolchain for Arm. (https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads).

Программа для отладки компилируется вместе с библиотекой GDB stub в составе одного проекта.

Чтобы добавить в свой проект режим отладки, необходимо:

1. Подключить в проект все Си файлы из папки GDB/src.
2. Прописать в настройках Keil путь к заголовочным файлам: ./GDB/inc.
3. Подключить в main.c библиотеку GDB: #include "gdb_stub.h".
4. В начале main() вызвать функцию StartGDB(), которая инициализирует и включает UART для отладки.
5. Добавить начальную точку останова, вызвав функцию gdb_bp().

Данная точка останова будет использоваться для входа в режим отладки (должна находится после включения UART).

6. Использовать startup.s из примера led_blink_debug.

После компиляции программы её образ в формате *.hex зашивается в память МК, используя Loader, а образ в формате *.axf передается на исполнение программе клиенту на стороне ПК. Весь процесс отладки происходит в командной строке Windows. Чтобы запустить командную строку необходимо нажать сочетание клавиш "Пуск"+R и в открывшемся окне ввести *"CMD"*.

Чтобы запустить скомпилированный образ программы *.axf в GDB клиенте, необходимо перейти в папку с клиентом. В команде используется путь по умолчанию (здесь и далее представлены команды, которые необходимо ввести в командную строку):

```
cd C:\Program Files (x86)\GNU Tools ARM Embedded\7 2018-q2-update\bin
```

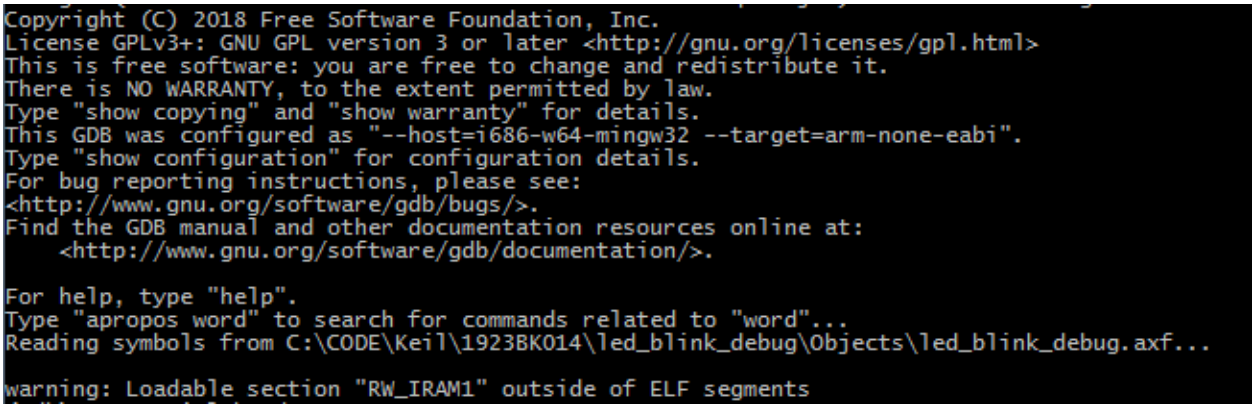
Вставка команды в консоль осуществляется кликом правой кнопкой мыши и выбором в открывшемся меню пункта "Вставить".

Чтобы запустить клиент с образом программы *.axf (данный файл находится в папке с проектом, на примере Keil: C:\...\project_folder\Objects\project_name.axf):

```
arm-none-eabi-gdb.exe C:\...\project_folder\Objects\project_name.axf
```

В названии папок должны использоваться только латинские буквы!

Если всё прошло успешно и путь указан верно, должно появиться следующее сообщение:



Теперь командная строка начинается с символов *(gdb)*, которые означают, что клиент GDB запущен, и все введённые команды обрабатываются им. Сначала необходимо указать рабочую папку проекта, так как по умолчанию задана: C:\Program Files (x86)\GNU Tools ARM Embedded\7 2018-q2-update\bin. Для этого необходимо воспользоваться командой *“cd <путь>”*. Проверить текущую рабочую папку можно командой *"pwd"*:

```
cd C:\CODE\Keil\1923BK014\led_blink_debug

pwd
```

Теперь необходимо задать настройки клиента на стороне ПК и установить соединение. В стандартном проекте скорость UART равна 115200, поэтому очень важно задать именно правильное значение:

```
set serial baud 115200
```

Далее необходимо установить соединение по COM порту с микроконтроллером. Перед данной операцией нужно обязательно закрыть программу Loader, которая также использует данное соединение:

```
target remote COM9
```

В случае успешного подключения должно появиться следующее сообщение, программа в этот момент находится в начальной точке останова, указанной функцией gdb_bp().

```
(gdb) set serial baud 115200
(gdb) target remote COM9
Remote debugging using COM9
0x00000164 in gdb_bp ()
(gdb)
```

Чтобы посмотреть листинг функции main введём команду:

```
list main
```

```
(gdb) list main
7       void LedOff(void);
8
9       int main()
10      {
11      /*---------- Start Up system ----------*/
12              int i = 0;
13
14              StartGDB();   // Enable GDB stub
15
16              /*---------- PortC settings for LEDS Line ----------*/
(gdb)
```

Однако, по умолчанию выводится только 10 строк кода. Чтобы продолжить вывод кода, необходимо либо ввести команду *"list"* либо нажать клавишу *Enter*. Можно также задать произвольное количество строк выводимого кода командой:

```
set listsize N
```

где N - количество строк.

Сразу после вывода кода можно указать точку останова, например, на строке 36, командой

```
break 36
```

Эту команду необходимо вводить только после вывода кода функции, т.к. номер строки 36 считывается именно из кода выведенной функции.

Чтобы запустить выполнение программы, выполним команду:

```
c
```

Дойдя до 36 строки, программа остановится, а в консоль будет выведено сообщение:

```
Breakpoint 1, main () at src\main.c:36
36                      LedOn();
(gdb)
```

Другие нужные команды для работы с точками останова:

```
info breakpoints  // Вывести в консоль все поставленные программно точки останова

delete breakpoints N  // Удалить N точку останова. Если ввести без аргумента N, то удалятся все точки останова
```

Чтобы выйти из режима отладки необходимо ввести команду:

```
detach
```

Выход из программы клиента GDB осуществляется командой:

```
quit
```

Полный набор команд приведён на официальном сайте GDB. (https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_toc.html)