

# ОТЛАДОЧНАЯ ПЛАТА RADIANT MIK32

Руководство по быстрому старту



17 ЯНВАРЯ 2025 Г. RADIANT.SU Г МОСКВА

# Оглавление

Введение	2
Системные требования	2
Краткое описание платы	3
Установка необходимого ПО	3
Установка драйвера WinUSB	3
Установка драйвера VCP FTDI	5
Варианты сборки проекта	11
Стратегия прошивки памяти программ	11
Первый запуск среды программирования	11
Запуск примера	12
Прошивка и отладка	14
Принцип запуска кода из SPIFI с максимальной производительностью	15
Прошивка кода без запуска отладчика	16
Проекты-шаблоны	16
Шаблоны из комплекта поставки	16
Как создать свой проект на базе шаблонов	16
Открытие проектов из комплекта примеров MikronIDE	17
Создание нового workspace	19

## Введение

Отладочная плата **RADIANT MIK32** предназначена для ознакомления с микроконтроллером K1948BK018 AДKБ.431290.418TУ (MIK32 Aмур), производства AO «Микрон». Данный продукт нацелен на помощь в изучении архитектуры микроконтроллера и не предназначена для встраивания в конечные устройства. На плате установлен JTAG отладчик и виртуальный последовательный порт, соединенный с выводами UART1. Данный продукт совместим с платами расширения стандарта ARDUINO-UNO. Для удобства работы с микроконтроллером все порты вводавывода продублированы на штыревые разъемы по краям платы.

#### Таблица 1 Основные сведения

Характеристика	Значение
Микроконтроллер	K1948BK018
Объем флэш памяти для приложения	8Мб, QSPI
Отладочный порт	JTAG
Интерфейс подключения платы	USB
Возможность программирования внешних схем	Да
Возможностью отделения от целевой платы	Да
Встроенный виртуальный СОМ порт для отладки	Да
Встроенный регулятор 3.3В	MIK1117
Порт 485	MIK3485
Разъем расширения ARDUINO-UNO	Да
Портов ввода-вывода на штыревых разъемах, шт	40 (все)
Кнопки управления	2
Светодиоды для отладки	2
Кварцевый резонатор высокочастотный	32МГц
Кварцевый резонатор низкочастотный	32768Гц
Габариты	68мм*93мм

# Системные требования

Для работы с микроконтроллерной платой потребуется следующее техническое оснащение:

- 1. Компьютер на базе Windows 10 или новее.
  - а. Права администратора, чтобы иметь возможность устанавливать новое ПО
  - b. Оперативная память не менее 8Гб
  - с. Свободное место на диске 50Гб
  - d. Свободный порт USB
- 2. Кабель с разъемом USB-type C

## Краткое описание платы



#### Рис. 1 Кратное представление отладочной платы и ее блоков

Для основной работы с платой достаточно простого подключения кабеля USB.

Питание резервного домена от батарейки (для питания часов реального времени RTC в режиме отключения основного питания) для плат версий 0.3 и ранее не предусмотрено.

Vprg – вход для подачи повышенного напряжения питания для программирования ОТР памяти 256 бит. Если эта память не используется, то оставить этот вход не подключенным. Для чтения этой памяти следует подавать 3.3В.

Джампер на разъеме XS14 "IDD" можно удалить и, подключив в разрыв амперметр, измерить потребление тока микроконтроллером (+ внешняя память).

# Установка необходимого ПО

Скачайте установочный файл <u>https://disk.yandex.ru/d/WVTSVBOD6Jj6nw</u> и установите среду разработки в папку C:\MikronIDE или в удобное место. Рекомендуется в корневую директорию C:\.

#### Установка драйвера WinUSB

Подключите плату к компьютеру кабелем USB. Дождаться пока автоматическая установка драйверов завершится.

Сначала рекомендуется удалить все драйвера



Рис. 2 Удаление автоматически установленных драйверов

Таким образом, чтобы в установленных устройствах в диспетчере устройств появилось два неопознанных компонента:



Рис. З Устаройста отладочной платы в Диспетчере устройств

Активируйте программу Zadig из "MikronIDE\tools\zadig-2.4.exe". Ссылка на последнюю версию: <u>https://zadig.akeo.ie</u>.

Zadig	- 🗆 X
<u>D</u> evice <u>Options</u> <u>H</u> elp	
Dual RS232-HS (Interface 1)         Driver       (NONE)         USB ID       0403       6010       01         WCID <sup>2</sup> X       Install Driver       Y	✓ ☐ Edit More Information WinUSB (libusb) libusb-win32 libusbK WinUSB (Microsoft)
5 devices found.	Zadig 2.4.721

Рис. 4 Установка драйвера для JTAG

Подключите плату. В меню Options нажмите на List All Devices. В выпадающем списке выберите устройство с названием:

#### Dual RS232-HS (Interface 1)

Затем проверьте что выбран один из совместимых драйверов, например, WinUSB или libusbK или libusb-win32 в белом поле ввода (выбирать кнопками со стрелками) и нажмите кнопку **Replace Driver** или **Install Driver**.

Для виртуальных машин Windows рекомендуется libusbK.

После установки **отключите** устройство и **подключите заново для начала работы**. Обновите список. На сером поле должен быть обозначен <u>новый</u> драйвер.

## Установка драйвера VCP FTDI

Установка виртуального СОМ-порта. На канал Dual RS232-HS (Interface 0) – нужно установить вручную драйвер



		$\times$
4	📱 Обновить драйверы — Dual RS232-HS	
	Как вы хотите провести поиск драйверов?	
	→ <u>Автоматический поиск драйверов</u> Windows выполнит поиск оптимального драйвера на компьютере и установит его на устройство.	
	→ Найти драйверы на этом компьютере Поиск и установка драйвера вручную.	
		Отмена

Рис. 5 Установка драйвера для VCP

		$\times$
÷	📕 Обновить драйверы — Dual RS232-HS	
	Поиск драйверов на этом компьютере	
	Искать драйверы в следующем месте:	
	C:\USB\FTDI_Drv	
	<ul> <li>Включая вложенные папки</li> <li>Выбрать драйвер из списка доступных драйверов на компьютере в этом списке перечисляются все доступные драйверы, совместимые с данным устройством а также прайверы аля устройств той же категории.</li> </ul>	
	устроиством, а также драиверы для устроиств той же категории.	
	<u>Да</u> лее Отмена	1

Рис. 6 Установка драйвера для VCP

Драйвер находится в папке tools/FTDI\_Drv. Новую версию можно скачать с сайта <u>https://ftdichip.com/</u>

После этого возникнет еще одно неопознанное устройство:



#### Рис. 7 После первого прохода установки

Нужно повторить процедуру установки FTDI\_Drv на это устройство.

В итоге должно образоваться два устройства в диспетчере устройств:

- Устройства HID (Human Interface Devices)
- Устройства USB
  - Dual RS232-HS
- Устройства безопасности

- У Контроллеры запоминающих устройств
- > 📃 Мониторы
- > Ш Мыши и иные указывающие устройства
- > 🚍 Очереди печати
- Порты (СОМ и LPT)
   USB Serial Port (СОМ10)
- Программные устройства
- > 🔲 Процессоры
- > 🖵 Сетевые адаптеры
- > 10 Системные устройства

Рис. 8 Диспетчер устройств после корректной установки всех драйверов

# Варианты сборки проекта

Есть три основных варианта работы кода (приложения пользователя):

- 1. Из оперативной памяти RAM
- 2. Из встроенной EEPROM
- 3. Из внешней QSPI FLASH памяти

Режим запуска выбирается с помощью джамперов на отладочной плате ВО и В1 (коммутирующие выводы микроконтроллера BOOT0-BOOT1 на 0V/3.3V) на разъеме CON2.

Запуск из ОЗУ удобен для проверки работы небольших блоков кода в рамках отладки отдельных небольших частей приложения или отладки функционирования периферийных устройств.

Работа основного приложения из EEPROM допускается. Данный подход ограниченно полезен в силу небольшого объёма EEPROM - 8кБ.

Работа «боевого» приложения во внешней памяти предполагается в качестве основного режима работы. Автономный запуск из внешней QSPI памяти (выводами контроллера BOOT0-BOOT1) возможен только при наличии резисторов подтяжки к линиям памяти D2 и D3. Штатная рекомендованная последовательность запуска приложения в QSPI памяти:

- запуск из встроенного EEPROM, конфигурация микроконтроллера и его интерфейсов
- переход (прыжок) в QSPI/SPIFI,

В этом случае подтяжки не обязательны.

Запуск из внешней памяти лишь установкой ВООТО-ВООТ1 рекомендуется использовать только для тестовых задач, так как включается этот интерфейс в базовой производительности:

- Без активации кэш памяти
- QSPI в MISO-MOSI режиме

В результате получается невысокая скорость работы всей системы.

## Стратегия прошивки памяти программ

Из-за различий видов памяти, из которых может работать приложение, запуск дебаггера разделен на две части:

- Загрузка кода в память с помощью скрипта mik32\_uploader
- Собственно вход в отладчик

Так же присутствует групповой запуск – вход в дебагер с предварительной загрузкой прошивки.

# Первый запуск среды программирования

Для запуска среды нажмите «Пуск» и найдите MikronIDE и выполните его

По умолчанию рабочее пространство проекта настроено на путь \MikronIDE\workspace. При первом запуске менять этот путь не нужно.

Разберем состав типового проекта (его файловую систему) на примере проекта "Bootloader":



Рис. 9 Базовый состав типового проекта



Рис. 10 Описание папки scripts

# Запуск примера

Выберите интересующий пример из окна проектов и двойным щелчком мыши откройте его



Рис. 11 Браузер проекта

#### Откройте файл main.c.

Далее следует выбрать конфигурацию сборки, варианты в примерах:

- Debug RAM
- Debug EEPROM
- Debug SPIFI (QSPI)

Конфигураций сборки может быть много, и они могут варьироваться помимо типа памяти, в которой будет работать код, степенью оптимизации и прочими опциями. Эти конфигурации можно создавать самостоятельно.

Для этого правой кнопкой мыши по проекту выбирается нужная конфигурация:

workspace	- Ec	lipse IDE										
<u>File Edit S</u> o	ource	e Refac <u>t</u> or <u>N</u> avigate Se	<u>a</u> rch <u>P</u> roject <u>R</u> un	<u>W</u> indo	ow <u>H</u> el	р						
s 🐐 🛙		🕸 Debug 🛛 🗸 💽	template_v2 Debug S	SPIFI	~ 🌣	: 📬 👻	BQ	🛞 🔻	≪ -	🔜 i 📮	<u>ي</u> ا	5 👩
陷 Project Exp	lorer	r×	🖻 🔄 🏹 🕴 🗖									
<ul> <li>Bootloa</li> <li>CoreMa</li> <li>freertos</li> <li>freentos</li> </ul>	der_v rk_v2 _v2 _	v2 2		>								
> 🐝 Binar		Go Into										
> 🔊 Inclu > 🔗 runtii > 🔗 src > 🍃 Debu		Open in New Window Show In Show in Local Terminal	Alt+Shift+W	>								
> 🗁 Debu > 🗁 Debu > 🗁 scrip 🗎 prg_e		Copy Paste Delete Source Move	Ctrl+C Ctrl+V Delete	>								
		Rename	F2									
	24 24	Import Export										
	Ł	Build Project Clean Project Refresh Close Project Close Unrelated Project	F5									
		Build Targets Index		>								
		Build Configurations		>	Set Act	tive	>	$\checkmark$	1 Deb	oug RAM		
		Profiling Tools		>	Manag	e			2 Deb	oug SPIF	5	
	0	Run As		>	Build A	AII		10.	3 Deb	ug_EEP	ROM vntace	ore-76

Рис. 12 Выбор конфигурации сборки RAM

#### После – можно собирать проект:



Рис. 13 Запуск загрузки и отладки

Или Ctrl+B.

Если сборка прошла без ошибок, о чем появится сообщение в консоли, то можно загружать прошивку в память и отлаживать код.

## Прошивка и отладка

Выберите конфигурацию, например, «Debug RAM» (как на скриншоте выше). Перемычками установите запуск из RAM.

Убедитесь, что курсор установлен на любой строке файла main.c. Если курсор будет установлен у другом окне, то отладка не запустится.

Запустите дебагер через панель инструментов:





Рис. 15 Запуск загрузки через панель инструментов

После чего запустится отладчик и можно отлаживать код, ставить точки останова, контролировать состояние переменных, памяти и т д.



Рис. 16 Отладчик

# Принцип запуска кода из SPIFI с максимальной

## производительностью

Принцип запуска кода из SPIFI следующий:



Рис. 17 Стратегия работы через загрузчик

В примерах есть проект Bootloader (стартовый загрузчик), который выполняет эти базовые функции. Загрузчик предварительно должен быть прошит в EEPROM микроконтроллера. После этого можно прошивать и отлаживать код в SPIFI на максимальной скорости. При этом сигналы BO-B1 должны быть всегда на выборе EEPROM. Bootloader можно доработать под выбранную микросхему памяти. А также добавить в него функционал обновления прошивки в QSPI памяти.

## Прошивка кода без запуска отладчика

Прошивка кода осуществляется через конфигурацию «внешний инструмент»: «External tool». Прошивальщик в SPIFI/EEPROM/RAM уже преднастроен в рабочем пространстве.

Для прошивки откройте интересующий проект. Выберите файл из этого проекта, например, main.c, установите курсор в этот файл. И в панели инструментов, нажав на иконку, запустите прошивку:

#### sr(mtvec, &\_\_TEXT\_START\_\_);

#### \_APB\_P\_CLEAR = PM\_CLOCK\_APB\_P\_GPI0\_1\_M

В результате код прошьется в память, на которую настроена конфигурация сборки, но отладчик не запустится.

# Проекты-шаблоны

#### Шаблоны из комплекта поставки

В рабочую среду включены несколько примеров-шаблонов:

- Bootloader. Работающий из встроенного EEPPROM его же можно назвать «загрузчиком». Так же в отладочных целях можно запустить проект из оперативной памяти. Код выполняет функции инициализации внешней памяти QSPI, включает кэш, и переводит исполнение кода на неё. Так же во внутреннюю EEPROM память можно поместить приложение для обновления внешней памяти.
- Template. Код, преимущественно рассчитанный на работу из внешней QSPI памяти он же - шаблон для «боевого приложения». Так же в конфигурацию этого проекта включены конфигурации на сборку в RAM и EEPROM. Для каких-то приложений может быть достаточно и этой небольшой памяти 8кБ, тогда внешняя память будет не нужна.
- 3. **CoreMark**. Проект для тестирования производительности микроконтроллера. Присутствуют конфигурации для QSPI, EEPROM и RAM.
- 4. **Template\_FreeRTOS**. Присутствуют конфигурации для QSPI, RAM. Для EEPROM объем выходного кода слишком большой, поэтому не рекомендуется.
- 5. SystemInfo. Данный пример при запуске на отладочной плате с подключенным часовым кварцевым резонатором позволяет определить калибровочный коэффициент для регистра калибровки HSI (в составе настроечного регистра WU->CLOCKS\_SYS). А так же данный пример вычитывает ID микроконтроллера и дату его производства. Вся информация выводится в UART 1 на скорости 9600N1.
- 6. **DMA\_UART** Пример с использованием DMA.

## Как создать свой проект на базе шаблонов.

Выбирите интеерсущий шаблон-проект, откройте его и правой кнопкой мыши по проекту выберите «Сору»:

눱 Project Explorer	× C/C++ Projects	∃ Æ \ 8    [
📁 amur_test_v2		
Bootloader_v2	2	
CoreMark_v2		
📁 DevBoardTest		
📁 DMA_UART		
> 👺 freertos_v2		
freertos_v	New	>
spifi_test_	Go Into	
📁 SystemInf	Open in New Window	
📁 template_	Show In	Alt+Shift+W >
📁 template_	Show in Local Terminal	>
🗀 TestingBo	Conv	Ctrl C
🖆 Maradiata C 🖳	Сору 🕟 Сор	V CIT+C

Далее по пустому полю нажмите правой кнопкой мыши и выберите «Paste»:

Ъ Project Explorer 🗵	C/C++ Projects	E 🕏 1	8	
📁 amur_test_v2				5
📁 Bootloader_v2				5
CoreMark_v2				E
📁 DevBoardTest				E
DMA_UART	New			>
> 😴 freertos_v2	Go Into			
freertos_v2_	On an in New Window			
📁 spifi_test_v0	Open in New Window			
📁 SystemInfo	Show In		Alt+Sh	itt+w>
template_v2	Show in Local Termina	l .		>
🗀 template_v2 🗎	Сору		0	Ctrl+C
🗀 TestingBoar 🛅	Paste 😓	Paste	C	Ctrl+V
	DIII			

И появится окно с выбором имени для нового проекта. После выбора имени – нажмите кнопку «Сору»

🛢 Сору	Project — 🗆 🗙
Project na	ame: freertos_v3
🗹 Use de	efault location
Location:	C:\MikronIDE\workspace\freertos_v3 Browse
	Choose file system: default \vee
?	Copy Cancel

Проект можно переименовать в будущем.

# Открытие проектов из комплекта примеров MikronIDE

В папке workspace содержатся проекты-примеры, которые по-умолчанию не добавлены в рабочее пространство. Для их добавления нужно «Открыть проект», сначала выбирается в меню функция «Open Project...»:

workspace - SystemInfo/src/main.c - Eclipse IDE

 Eile
 Edit
 Source
 Refactor
 Navigate
 Search
 Project

 New
 Alt+Shift+N >
 Open File...
 Alt+Shift+N >
 Image: Comparison of the system in the sys

Далее выбирается папка, где находится интересующий проект, после чего следует нажать на кнопку Finish.

Import Projects from File System or Archiv	/e		_		
Import Projects from File System or A	rchive				5
This wizard analyzes the content of your fold	ler or archive file to	find projects and im	port them in the I	DE.	
Import source: C:\1\MikronIDE\workspace\	CoreMark	~	Directory	Archive	
Close newly imported projects upon comp Use installed project configurators to: Search for nested projects Detect and configure project natures	oletion				
Working sets			Γ	New	
Working sets:			~	Select	
?	< Back	Next >	Finish	Cancel	

После этого проект будет добавлен в среду разработки и с ним можно работать:



# Создание нового workspace

При создании нового workspace, основная проблема — в новом workspace отсутствуют конфигурации загрузки прошивки и запуска дебагера

Поэтому из шаблонного workspace сначала следует экспортировать все рабочие «конфигурации»:

Export —	
Select Export launch configurations to the local file system.	Ż
<u>S</u> elect an export wizard:	
type filter text	
<ul> <li>&gt; ➢ C/C++</li> <li>&gt; ➢ Install</li> <li>∽ ➢ Run/Debug</li> </ul>	^
<ul> <li>Breakpoints</li> <li>Launch Configurations</li> </ul>	
> > Team	
> 🗁 Other	~
? < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

Сохраняем на диске и запоминаем путь.

Export Launch Configurations —		×
Export Launch Configurations		(A)
Select launch configurations to export		
Launch Configurations:		
> C/C++ Application		
> 🗹 🖻 GDB OpenOCD Debugging		
> 🔽 🖪 Launch Group		
> _ Ya Program		
Select All Deselect All		
Location: C:/MikronIDE/configurations	Brow	s <u>e</u>
<u>O</u> verwrite existing file(s) without warning		
Image: Second system     Mext >     Einish	Canc	el

#### После, создаем папку для нового workcpase

## workspace - template/src/main.c - Eclipse IDE

<u>File Edit Source Refactor Navigate Search Project Run Window H</u>elp

	New	w Alt+Shift+N > 🗸 🔅 🖻				s 🔹 🗖
	Open File Open Projects from File System		° – –	🔏 m	nain.c × 🗟 uart_lib.c	<u></u> в п
_	Recent Files	>		52 53	// extint on	user
	Close Editor	Ctrl+W		54	GPIO_1->DIRE	CTION
	Close All Editors	Ctrl+Shift+W		55	11 :	
	Save	Ctrl+S		50	GPTO TRO->  T	gene NF MU
	Save As			58	GPIO IRQ->ED	GE = :
B	Save All	Ctrl+Shift+S		59	GPIO_IRQ->LE	VEL_C
	Revert			60	GPIO_IRQ->EN	ABLE_
	Move			61	// interrunt	raca
P	Rename	F2		63	EPIC->MASK L	EVEL
8	Refresh	F5		64	_	_
	Convert Line Delimiters To	>		65	// global in	terru
۵	Print	Ctrl+P		66	set_csr(mstates) set_csr(mie.	tus, I MIE I
2	Import			68	,	
4	Export			69	while (1)	
	Properties	Alt+Enter		70	{	
_				/1	GP10_0->0	DOLLO
	Switch Workspace	>	{eclipse_home}///workspace			
	Restart		Othry			
_	Exit					
				1		×

## Выбираем папку для нового workspace:

TRO->ENARLE SET = 1 << 7	// enable 7th line X	
Select a directory as workspace Clipse IDE uses the workspace directed t	e ctory to store its preferences and development artifacts.	
Vorkspace: \${eclipse_home}//./w	orkspace ~ Browse	
ni → Recent Workspaces	Select Workspace Directory	X
<pre>\${eclipse home}///workspace</pre>	← → ч ↑ 🍠 Этот компьютер >	
<ul> <li>Copy Settings</li> </ul>	Упорядочить 🔻	<u>∎</u> r • <b>(</b> )
× ⑦	<ul> <li>Этот компьютер</li> <li>Папки (7)</li> </ul>	^
<pre>\R \RT_WaitTransmission(UART_1);</pre>	> 🔚 Видео > 🛅 Документы	Документы
🖉 Tasks 🖳 Console 🔲 Properties 🔊	<ul> <li>→ Загрузки</li> <li>&gt; Музображения</li> <li>&gt; Музыка</li> </ul>	Изображения
	<ul> <li>Э Объемные объ</li> <li>Вабочий стол</li> </ul>	Объемные объекты
2	> 😃 Локальный дисі Рабочий стол	
2	> 😪 Мікгоп (М:) Устройства и диски (1)	
	Папка: Этот компьютер	
		Выбор папки Отмена

Далее открываем ново-созданный «пустой» workspace и импортируем сохраненные настройки

Через File→Import...

Import				×
Select Import launch configurations from the l	local file system.		[	2
Select an import wizard:				
type filter text				×
<ul> <li>&gt; &gt; Oomph</li> <li>&gt; &gt; RPM</li> <li>&gt; &gt; Run/Debug</li> <li>&gt; Breakpoints</li> <li>&gt; Launch Configurations</li> <li>&gt; &gt; Team</li> <li>&gt; &gt; TextMate</li> </ul>				^
> 🗁 Tracing > 🗁 XML				~
? < Back	Next >	Finish	Canc	el

Import Launch Configurations	_		×				
Import Launch Configurations Import launch configurations from the local file system							
From Directory: C:/MikronWORK		Brow	/s <u>e</u>				
<ul> <li>✓ ■ ➢ MikronWORK</li> <li>☑ ➢ Configurations</li> <li>&gt; □ ➢ MikronIDE</li> </ul>	Code Load.launch	ch					
Overwrite existing launch configurations without warning.							
? < <u>B</u> ack	lext > <u>F</u> inish	Cano	cel				

#### Конфигурации теперь на своих местах, Все готово к работе!

