

# USB JTAG adapter

Руководство пользователя.

Версия 5.0.1



JetLink5 – USB to J7AG adapter

#### Краткое описание

*Jetlink* – это JTAG эмулятор, подключаемый по интерфейсу USB к компьютеру и работающий под управлением Windows 2000, Windows XP, Windows 2003 или Windows Vista.

#### Возможности:

- Интерфейс USB 2.0 Full speed.
- Поддержка всех ARM7/ARM9 микроконтроллеров, а также Cortex M3, включая Thumb режим.
- Скорость загрузки до 720 Кбайт/сек.
- Не требует отдельного источника питания.
- Максимальная скорость JTAG 12 МНz.
- Автоматическое распознавание типа ядра.
- Автоматическое определение скорости.
- Поддерживает отладку устройств в цепочке (JTAG chain).
- Диапазон напряжения интерфейса 1,2 3,3В.
- Поддерживает адаптивное тактирование.

#### Комплектация:

- USB JTAG адаптер (далее Jetlink)
- USB кабель А-В
- 14 или 20 контактный шлейф для подключения к отладочной плате.
- CD диск.
- Упаковочная коробка.

#### Первое включение.

Запустите программу Setup\_JlinkARM находящуюся на диске.









## JetLink5—USB to JTAG adapter



👼 Installation Co	nplete 🔀
₩ *	J-Link ARM V3.20h has been successfully installed. Press the Finish button to exit this installation.
	< Back Finish > Cancel

На рабочем столе появятся ярлыки быстрого запуска.



После завершения установки программы Вы обнаружите в папке Program Files/Segger следующее:

- Jlink.exe простая утилита для тестирования адаптера.
- JlinkTCPIPserver.exe утилита для удаленной отладки.
- JFlashARM.exe программа для программирования ARM микроконтроллеров и внешней памяти.
- JMem.exe простая утилита для просмотра памяти.
- Папку USBDriver, в которой содержится драйвер для работы устройства.
- Папку Samples папка содержащая примеры проектов для JFlash и RDI (Remote Debugging Interface)
- Папку Doc содержащую документацию на английском языке.

### Примечание:

Свежую версию программы Setup\_JlinkARM Вы всегда можете скачать с сайта <u>http://www.segger.com/download\_jlink.html</u>

После того как программа установлена, подключите устройство к компьютеру. ОС Windows обнаружит новое устройство



и предложит установить для него драйвера.

JetLink5 – USB to J7AG adapter

Мастер обновления оборудования	
	Мастер обновления оборудования Этот мастер помогает установить программное обеспечение для указанного устройства: J-Link driver
	Если с устройством поставляется установочный диск, вставьте его.
	Выберите действие, которое следует выполнить. О Автоматическая установка (рекомендуется)
	Для продолжения нажмите кнопку "Далее".
	< Назад Далее > Отмена

Выберите «Установка из указанного места» и укажите путь к папке USBDriver (обычно это Program Files/Segger/JLinkARM\_Vxxx/ USBDriver.)

Мастер обновления оборудования		
Задайте параметры поиска и установки.		
Выполнить поиск наиболее подходящего драйвера в указанных местах.		
Используйте флажки для сужения или расширения области поиска, включающей по умолчанию локальные папки и съемные носители. Будет установлен наиболее подходящий драйвер.		
🗌 Поиск на сменных носителях (дискетах, компакт-дисках)		
Включить следующее место поиска:		
C:\Program Files\SEGGER\JLinkARM_V360a\USB 🔽 🛛 🛛 🛛 🗤 🖉		
Не выполнять поиск. Я сам выберу нужный драйвер.		
Этот переключатель применяется для выбора драйвера устройства из списка. Windows не может гарантировать, что выбранный вами драйвер будет наиболее подходящим для имеющегося оборудования.		
< Назад Далее > Отмена		

После установки драйверов устройство готово к работе. Возможно ОС Windows потребует перезагрузить компьютер, выполните это требование.



Устройство установлено.

JetLink5 – USB to J7AG adapter

#### Тестирование устройства.

С помощью программы Jlink.exe Вы можете протестировать устройство и измерить скорость работы эмулятора. Для этого подключите устройство к компьютеру, подключите к эмулятору микроконтроллер и запустите программу Jlink.exe.

Вы должны увидеть нечто похожее:



Если имеется сообщение об ошибке – не найдено ARM ядро, проверьте правильность подключения микроконтроллера к адаптеру и наличие питания.

Вы можете проверить работоспособность адаптера с помощью следующих команд:

f	Firmware info
h	halt
g	Go
S	Single step the target chip
st	Show hardware status
mem	Show memory. Syntax: mem <addr>, <numbytes> (hex)</numbytes></addr>
wl	Write 8 bits. Syntax: w1 <addr>, <data>(hex)</data></addr>
w2	Write 16 bits. Syntax: w2 <addr>, <data>(hex)</data></addr>
w4	Write 32 bits. Syntax: w4 <addr>, <data>(hex)</data></addr>
wm	Write test words. Syntax: wm <numwords></numwords>
is	Identify length of scan chain select register
ms	Measure length of scan chain. Syntax: ms <scan< td=""></scan<>
	chain>
q	Quit
r	Reset target (RESET)
rx	Reset target (RESET). Syntax: rx
	<delayafterreset></delayafterreset>
RSetType	Set the current reset type. Syntax: RSetType <type></type>
Regs	Display contents of registers
SetBPA	Set ARM breakpoint. Syntax: SetBPA <addr></addr>
SetBPT	Set Thumb breakpoint. Syntax: SetBPT <addr></addr>
ClrBP	Clear breakpoint. Syntax: ClrBP <bp_handle></bp_handle>
loadbin	Load binary file into target memory. Syntax: loadbin <filename>, <addr></addr></filename>

~		
SetPC	Set the PC to specified value. Syntax: SetPC <addr></addr>	
le	Change to little endian mode	
be	Change to big endian mode	
log	Enables log to file. Syntax: log <filename></filename>	
Ice	Show state of the embedded ice macrocell (ICE breaker)	
wi	Write Ice reg. Syntax: wi <reg>, <data>(hex)</data></reg>	
etm	Show ETM statusre Read ETM reg. Syntax: re <reg></reg>	
we	Write ETM reg. Syntax: we <reg>, <data>(hex)</data></reg>	
es	Start trace	
TAddBranch	TRACE - Add branch instruction to trace buffer. Paras: <addr>,<baddr></baddr></addr>	
TAddInst	TRACE - Add (non-branch) instruction to trace	
TCLOOK	TDACE Clear buffer	
	TRACE - Clear Duller	
TSetSize	TRACE - Set Size of trace builter	
TSELFORMAL	TRACE - Selformal	
ISR	TRACE - Show Regions (and analyze trace buller)	
TStart	TRACE - Start	
TStop	TRACE - Stop	
thg	Run go/halt 1000 times	
ts	Run step 1000 times	
testwspeed	Test download speed. Syntax: testwspeed [ <addr>]</addr>	
testrspeed	Test upload speed. Syntax: testrspeed [ <addr>]</addr>	
Config	Set number of IR and DR bits before ARM device. Syntax: Config <ir> &lt; DR&gt;</ir>	
	Syntax: Config <ir> &lt; DR&gt;</ir>	
speed	Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000</khz></ir>	
speed speed a	Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking.</khz></ir>	
speed speed a i	Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU)</khz></ir>	
speed speed a i wjc	Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex)</data></khz></ir>	
speed a i wjc wjd	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec)</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110)</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (11110) Reset TAP Controller (nTRST)</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0 t1	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0 t1 t0 t1 t1 t1	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0 t1 t1 trst0 trst1	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST Set TRST</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 1 t0 t1 trst0 trst1 usb	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST Set TRST Connect to J-Link via USB Syntax: usb contax</numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 1 t0 t1 t1 trst0 trst1 usb	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST Set TRST Connect to J-Link via USB. Syntax: usb <port>, where port is 0 3</port></numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0 t1 t1 trst0 trst1 usb	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST Set TRST Connect to J-Link via USB. Syntax: usb <port>, where port is 03 Connect to J-Link Server via TCP/IP</port></numbits></data32></data></khz></ir></pre>	
speed a i wjc wjd RTAP rt c00 c 0 1 t0 t1 trst0 trst1 usb ip wconf	<pre>Syntax: Config <ir> &lt; DR&gt; Set fixed JTAG speed in kHz. Syntax: speed <khz>, e.g. speed 2000 Use adaptive clocking. Read JTAG Id (Host CPU) Write JTAG command (IR). Syntax: wjc <data>(hex) Write JTAG data (DR). Syntax: wjd <data32>(hex), <numbits>(dec) Reset TAP Controller using state machine (111110) Reset TAP Controller (nTRST) Create clock with TDI = TMS = 0 Clock Clear TDI Set TDI Clear TMS Set TMS Clear TRST Set TRST Connect to J-Link via USB. Syntax: usb <port>, where port is 03 Connect to J-Link Server via TCP/IP Write configuration byte Syntax: wronf</port></numbits></data32></data></khz></ir></pre>	

rconf	Read configuration bytes. Syntax: rconf
usbaddr	Configures the usb address of connected J-Link:
	Syntax: usbaddr = <addr></addr>

К примеру, набрав цифру «1» выход TDI эмулятора примет уровень логической 1. Таким образом, можно легко обнаружить неисправность в соединении адаптера с микроконтроллером. С помощью команды «speed» Вы можете задать скорость, а с помощью «testspeed» Вы можете проверить скорость:



## Использование Jetlink совместно со средой IAR EW.

Запустите IAR, откройте проект. Выберите Project -> Options.

eneral Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Simulator Angel IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI Third-Party Driver	Target     Output     Librar       Processor variant     Core     ARM7       Core     ARM7       Device     Atmel x       FPU     None       ✓     Generate interwork       Processor mode     Arm       ✓     Thumb	v Configuration   Library	options MISRA C	
---	--	---------------------------	-----------------	--

JetLink5 – USB to J7AG adapter

Выберите строку «Debugger». В выпадающем меню «Driver» выберите пункт J-Link/J-Trace.

ategory:	Factory Settings
eneral Options	
C/C++ Compiler	Setup Download Extra Options Plugins
Assembler	
Custom Build	Driver 🔽 🔽 Run to
Build Actions	Simulator main
Debugger	Simulator
Simulator	IAR ROM-monitor
Angel	J-Link/J-Trace
IAR ROM-monitor	Macraigor
J-Link/J-Trace	\$PROJ_DIR\$\resource\\$AM7_RAM.mac
Macraigor	- Device description file
RDI	
Third-Party Driver	
	\$TOOLKIT_DIR\$\CONFIG\ioAT91SAM7S256.ddf

Теперь выберите строку «J-Link/J-Trace. Здесь пока можно ничего не менять. Если, используется отладка на локальном компьютере, в закладке Connection установите USB. Для удаленной отладки выберите TCP/IP и введите IP адрес удаленного компьютера, через который будет производиться отладка.

Category:		Factory Settings
C/C++ Compiler	Setup Connection Breakpoints	
Assembler	Communication	
Custom Build		
Linker	C TCD/ID are bbb coo ddd	
Debugger		
Simulator	JTAG scan chain	
Angel	□ JTAG scan chain with multiple targets	
IAR ROM-monitor		
LMLETDI		
Macraigor	Contains non-ARM devices	
RDI	Preceeding bits:	
Third-Party Driver		
	\$TUULKIT_DIH\$\cspycomm.log	

Закройте окно Options, нажав кнопку OK.

Теперь Вы можете, нажав кнопку Debug запустить проект на компиляцию и загрузку в ОЗУ/Flash микроконтроллера. Если в проекте нет ошибок - Вы увидите индикатор прогресса загрузки.

Progress	$\sim$
Programming flash	

#### Использование Jetlink совместно со средой Keil uVision.

Запустите Keil. Откройте Ваш проект.



### Зайдите в Настройки проекта (Project | Options)

Options for Target 'LPC2378 Flash'	×
Device Target Output Listing User C/C++ Asm Li	nker Debug Utilities
C Use Simulator Settings	Use: RDI Interface Driver Settings ULINK ARM Debugger
Load Application at Startup     Run to main()     Initialization File:      Edit	ULINK Cortex-M3 Debugger Load Interface Driver initializatiq Signum Systems JTAGjet Edit
Restore Debug Session Settings     Breakpoints     Vatchpoints & PA     Memory Display	Restore Debug Session Settings           Breakpoints         Toolbox           Watchpoints         Memory Display
CPU DLL: Parameter: D SARM.DLL -cLPC2100	river DLL: Parameter:
Dialog DLL: Parameter: D DARMP.DLL pLPC2378	ialog DLL: Parameter: ARMP.DLL pLPC2378
OK Cance	el Defaults Help

JetLink5 – USB to J7AG adapter

Перейдите на вкладку **Debug** 

Options for Target 'LPC2378 Flash'	×	
Device Target Output Listing User C/C++ Asm	Linker Debug Utilities	
O Use Simulator Settings	⊡se: RDI Interface Driver     ✓ Settings	
Limit Speed to Real-Time	ULINK ARM Debugger	
Load Application at Startup     Run to main() Initialization File:	Load Luminary Eval Board Initializatiq Signum Systems JTAGjet	
Bestore Debug Session Settings	Bestore Debug Session Settings	
	Reakpoints I Toolbox	
✓ Watchpoints & PA	Vatchpoints	
Memory Display	Memory Display	
	I	
CPU DLL: Parameter:	Driver DLL: Parameter:	
SARM.DLL -cLPC2100	SARM.DLL	
Dialog DLL: Parameter:	Dialog DLL: Parameter:	
DARMP.DLL -pLPC2378	TARMP.DLL ·pLPC2378	
OK Ca	incel Defaults Help	

Выберите RDI Interface driver. Нажмите кнопку **Settings**, расположенную рядом. Укажите путь к JLinkRDI.dll (Обычно находится в Program File/Segger/JLinkARM\_Vxxx/

RDI Interface Driver Setup		X
Browse for RDI Driver DLL		
C:\Program Files\SEGGER\JLinkAR	M_V359a\JLinkRDI.dll	
Browse for ToolConf File		
Debug Cache Options Cache Code Cache Memory	Configure BDI Driver	
	OK Cancel	Help

Нажмите «Configure RDI Driver». Откроется диалог настройки RDI

General Init JTAG Flash B	? ×         reakpoints       CPU       Log         J-Link-RDI is an RDI compliant software for J-Link         ARM. It requires a license (RDI), which can be obtained from SEGGER (www.segger.com).         This software is also capable of programming the flash memory of several ARM micros, which can be used to download you program to flash (Requires the add. license "FlashBL") and to set an unlimited number of software breakpoints in flash (Requires the add. license "FlashBP").         Connection to J-Link         © USB       Device 0         © LICP/IP
C:\Program Files\SEGGER\JLink	About License <u>RM_V359aVLinkRDI.ini</u> <u>Beset Config</u>

Примечание: Скорее всего, в данный момент Вы увидите окно с просьбой ввести номер лицензии. Подробнее о лицензии смотрите в пункте «Лицензии».

После завершения настройки, откомпилируйте проект (**Project** | **Build Target**) и запустите отладку (**Debug** | **Start/Stop debug session**).

₩ Blinky - µVision3 - [Disassembly]		
R Eile Edit View Project Debug Flash Periph	erals <u>I</u> ools <u>S</u> VCS <u>W</u> indow <u>H</u> elp	>
🏠 😂 🖬 🗿 👗 🖻 🛍 🏼 으 오 🎼	律 ル % % % % <b>%</b>	<u>■</u> # # + → 12 @ @ <b>@ ■</b> # # # #
🔐 🖹 😫 🖓 🔂 ( <sup>3</sup> + 0) → ½± 0± [	Q 🔉 🖤 ¥ 🗖 E 🔤 🖬 🦻 🗡	
Project Workspace v x	332: Vectors LDR	PC, Reset Addr
Register Value	➡ 0x 00000000 E59FF 018 LDR	PC,[PC,#8x8018]
E Current		PC, Undef_Addr
R0 0x0000000	334: LDR	PC, SWI Addr
R1 0x0000000	0x00000008 E59FF018 LDR	PC,[PC,#0x0018]
R2 0x0000000	335: LDR	PC, PAbt_Addr
	0x0000000C E59FF018 LDR	PC,[PC,#0x0018]
R5 0x0000000	330: LUK 8y88888818 E59EE818 LDR	PC FPC #8y88181
R6 0x0000000	337: NOP	; Reserved Vector
R7 0x0000000	338:; LDR	PC, IRQ_Addr
R8 0x0000000	0x00000014 B9206E50 STHLTD	B R01, {R4, R6, R9-R11, R13-R14}
R9 0x0000000	339: LUK 8y88888818 F51FF128 LDR	PC, [PC, #-0X0120] ; Vector from VicVectHoor PC [PC #-8x8128]
	340: LDR	PC, FIQ Addr
	341:	
	342: Reset_Addr DCD	Reset_Handler
	343: Undef_Haar DCD 366- SWI Addr DCD	Undet_Handler SWI Wandler
R15 (PC) 0x0000000	345: PAbt Addr DCD	PAbt Handler
E CFSN 0x00000003	346: DAbt_Addr DCD	DAbt_Handler
⊞	347: DCD	Ø ; Reserved Address
E-Fast Interrupt	348: INQ_HUUF DCD 349: FIO Addr DCD	FIO Handler
E interrupt		···•
E F E U B V F V Te	LPC2300.s Blinky.c ELCD_4	bit.c 📄 IRQ.c 📄 Abstract.txt 🕵 Disassembly
Load "C:\\Keil303o\\AR	M\\RV30\\Boards\\Keil\\I	MCB230 Address: 0x3fffc000
		0x2FFFC000, 0000000 0000000
		0x3FFFC000: 0000000 0000000
		0x3FFFC008: 0000000 0000000
		0x3FFFC010: 0000000F 00000000
		0x3FFFC010: 0000000 0000000
		0x3FFFC020: 0000000F 00000000
		UK3FFFC030: 0000000 0000000
ASSIGN BreakDisable Br	eakEnaple BreakKill	
811		E d b Memory #1 / Memory #2 / Memory #3 / Memor
For Help, press F1		RDI Interface Driver t1: 0.00000000 sec

Настройка загрузчика Flash памяти через интерфейс RDI.

Примечание: Эта функция требует лицензии. Подробнее о лицензии смотрите в пункте «Лицензии».

Запустите Keil. Откройте Ваш проект. Зайдите в Настройки проекта (Project | Options).

ngs Update Target before Debugging
Edit

Нажмите «Settings» и выберите «J-Link Flash Programmer» в выпадающем списке. Нажмите «OK».

## Настройки RDI интерфейса.

J-Link RDI Configuration General Init JTAG Flash B	reakpoints       CPU       Log         J-Link-RDI is an RDI compliant software for J-Link         ARM. It requires a license (RDI), which can be obtained from SEGGER (www.segger.com).         This software is also capable of programming the flash memory of several ARM micros, which can be used to download your program to flash (Requires the add. license "FlashDL") and to set an unlimited number of software breakpoints in flash (Requires the add. license "FlashBP").
Location of gonfig file	Connection to J-Link
	OK Cancel Apply

Connection to J-Link - способ подключения, USB или Ethernet.

**About** – здесь Вы можете ввести номер лицензии. Подробнее – в разделе Лицензии.

**Config file** – использование конфигурационного файла позволяет легко менять настройки для, например, разных микроконтроллеров.

JetLink5 – USB to J7AG adapter

#### INIT

J-Link RDI Configuration	? ×
General Init JTAG Flash Bre	akpoints CPU Log
	OK Cancel Apoly
	Cancer Apply

### Macro file

Макро файл может быть использован для загрузки пользовательских настроек, чтобы настроить Jlink RDI со специальными командами для отдельных микросхем или операций. Например, макро файл может быть использован для инициализации целевой системы только в коей мере необходимости.

Команда	Описание	
SetJTAGSpeed(x);	Sets the JTAG speed, $x =$ speed in kHz (0=Auto)	
Delay(x);	Waits a given time, x = delay in millisecond s	
Reset(x);	Resets the target, $\mathbf{x}$ = delay in milliseconds	
Go();	Starts the ARM core	
Halt(); Table 3.1: Macro file commands	Halts the ARM core	
Read8(Addr);		
Read16(Addr);	Reads a $8/16/32$ bit value, Addr = address to read	
Read32(Addr);		
<pre>Verify8(Addr, Data);</pre>	Verifies a 8/16/32 bit value, Addr = address to verify	
<pre>Verify16(Addr, Data);</pre>	(as hex value) Data = data to verify (as hex value)	
<pre>Verify32(Addr, Data);</pre>		
Write8(Addr, Data);	Writes a 8/16/32 bit value, Addr = address to write	
Write16(Addr, Data);	(as hex value) Data = data to write (as hex value)	
Write32(Addr, Data);		
WriteVerify8(Addr, Data);	Writes and verifies a 8/16/32 bit value, Addr =	
WriteVerify16(Addr, Data);	address to write (as hex value) Data = data to write	
WriteVerify32(Addr, Data);	(as hex value)	
WriteRegister(Reg, Data);	Writes a register	

· · · · · · · · · · · · · · · · · · ·	
WriteJTAG_IR(Cmd);	Writes the JTAG instruction register
WriteJTAG_DR(nBits, Data);	Writes the JTAG data register

Пример макро файла.

## JTAG

J-Link RDI Configuration	? ×
General Init JTAG F	- Flash Breakpoints CPU Log
ITAG speed	
C Auto selection	
C Adaptive clocking	
⊙ 30 ▼ kHz	
🖵 🗖 JTAG scan chain with	n multiple devices
Position 0	[Rilen 0
0 is closest to TDI.	Sum of IRLens of devices closer to TDI.
	Inden of Anim onlys is 4.
	Verify JTAG config
	OK Cancel Apply

### **JTAG speed**

Доступны три варианта настройки скорости –

- Fixed JTAG speed
- Automatic JTAG speed
- Adaptive clocking

#### **Fixed JTAG speed**

Target используется фиксированная скорость. Максимальная скорость зависит от конкретного микроконтроллера. В обычных ARM микроконтроллерах без логики JTAG синхронизации (такие как ARM7-TDMI) может работать на скоростях самого микроконтроллера. ARM микроконтроллеры с логикой синхронизации (такие как ARM7-TDMI-S, ARM946E-S, ARM966EJ-S) могут работать на скоростях до 1/6 от скорости самого микроконтроллера. Установка скорости более 10 МГц не рекомендуется.

#### Automatic JTAG speed

Максимальная скорость JTAG устанавливается автоматически. Для микроконтроллеров без логики синхронизации работа в данном режиме может быть нестабильна.

#### **Adaptive clocking**

Если микроконтроллер поддерживает RTCK сигнал, выберите адаптивное тактирование для синхронизации тактирования процессора внутри с внешним тактированием. Это гарантируем полное отсутствие проблем с синхронизацией в JTAG.

#### JTAG scan chain with multiple devices

JTAG scan chain позволяет указывать регистр (Instruction Register) в целевой системе. Это может потребоваться, если несколько устройств находятся в системе.

## **Flash configuration**

J-Link RDI Configuration 🔹 🔀
General Init JTAG Flash Breakpoints CPU Log
C Enable flash programming
Allows programming the flash. This is required to download a program into flash memory or to set software breakpoints in flash (flash breakpoints).
Device Atmel AT91SAM7S64 Clock speed 48000000 Hz
RAM 16 KB @ address 0x200000
Flash 64 KB @ address 0x100000
Flash is mirrored @ address 0x0
Cache flash contents Allows caching of flash contents. This avoids reading data twice and speeds up the transfer between debugger and target.
Allow flash <u>d</u> ownload Allows program download to flash. Your debugger does not need to have a flash loader. This feature requires an additional license (FlashDL).
☑ Show info window during download
OK Cancel Apply

Enable flash programming – разрешает программирование flash. Это необходимо для загрузки флеш памяти или использования flash breakpoints.

Если вы используете данную функцию, необходимо правильно указать тип микросхемы, базовый адрес флеш памяти и скорость тактирования.

#### **Cache flash contents**

Если разрешено, то содержимое флеш кешируется, чтобы избежать двойного чтения данных и повысить скорость обмена между эмулятором и микроконтроллером.

#### Allow flash download

Функция программному обеспечению RDI загружать флеш. Небольшой фрагмент кода будет загружен и запущен на выполнении в ОЗУ микроконтроллера, который затем будет программировать флеш память. Это обеспечивает возможность оперативной загрузки отладчиком, даже без наличия встроенного флеш загрузчика.

По окончанию загрузки будет отображено информационное окно.

J-Link flash programming		
	Finished flash programming.	

## **Breakpoints**



#### **Use software breakpoints**

Функция позволяет устанавливать неограниченное число точек прерывания, если программа находится в оперативной памяти, установка и переустановка точек прерывания в зависимости от программного кода.

#### Use flash breakpoints

Функция позволяет устанавливать неограниченное число точек прерывания, если программа находится в ОЗУ или во флэш-памяти путем сброса точек прерывания и установки заново в зависимости от программного кода.

JetLink5 – USB to J7AG adapter

#### Информационное окно отображает текущую операцию.

J-Link flash programming

Programming sector 0 (128 Bytes @ addr 0x00000000)

### CPU

J-Link RDI Configuration ? 🗙
General Init JTAG Flash Breakpoints CPU Log
Allows the emulator to simulate individual instructions when single stepping instructions. This does not normally have any disadvantages and makes debugging much faster, especially when using flash breakpoints.
Reset strategy
J-Link supports different reset strategies. This is necessary because there is no single way of resetting and halting an ARM core before it starts to execute instructions.
Hardware, halt after reset (normal)
The hardware RESET pin is used to reset the CPU. After reset release, J-Link continuously tries to halt the CPU. This typically halts the CPU shortly after reset release; the CPU can in most systems execute some instructions before it is halted. The number of instructions executed depends primarily on the JTAG speed; the higher the JTAG speed, the faster the CPU can be halted. Some CPUs can actually be halted before executing any instruction, because the start of the CPU is delayed after reset release. If a pause has been specified, J-Link waits for the specified time before trying to halt the CPU. This can be useful if a bootloader which resides in flash or ROM needs to be started after reset.
OK Cancel Apply

#### **Reset strategy**

Определяет, как J-Link RDI определяет запросы на сброс от программы. J-Link сбросить поддерживает различные варианты сброса. Это необходимо потому, что нет единого способа сброса и остановки ARM ядра, перед стартом и началом выполнения инструкция.

Исполнения инструкций после сброса может привести к различным проблемам. Некоторые ARM микроконтроллеры не могут быть до конца "остановлены", это значит, что они не могут переключаться в режим отладки (CPU невозможно остановить).

Доступные способы сброса:

- Hardware, halt after reset (normal)
- Hardware, halt after reset using WP
- Hardware, halt after reset using DBGRQ
- Hardware, halt with BP@
- Software, for Analog Devices ADuC7xxx MCUs
- No reset

#### Hardware, halt after reset (normal)

Вывод аппаратного сброса используется для сброса процессора. После выполнения сброса J-Link непрерывно пытается остановить процессор. Это обычно останавливает CPU вскоре после сброса; процессор может в большинстве систем выполнить ряд некоторых инструкций, прежде чем он будет остановлен. Число выполненных инструкций зависит главным образом от скорости JTAG: чем быстрее скорость JTAG, тем быстрее CPU может быть остановлен. Некоторые процессоры могут быть остановлены перед выполнением инструкций, потому что имеется задержка после сброса. Если задана задержка, J – Link будет ждать заданное время, прежде чем пытаться остановить процессор. Это может быть полезно, для старта загрузчика, который находится в флэшпамять или ROM, которые запускаются после перезагрузки.

#### Hardware, halt after reset using WP

Вывод RESET используется для сброса процессора. После сброса J-link непрерывно пытается остановить процессор. Это обычно останавливает CPU вскоре после сброса; процессор может в большинстве систем выполнить ряд некоторых инструкций, прежде чем он будет остановлен. Число выполненных инструкций зависит главным образом от скорости JTAG: чем быстрее скорость JTAG, тем быстрее CPU может быть остановлен. Некоторые процессоры могут быть остановлены перед выполнением инструкций, потому что имеется задержка после сброса.

#### Hardware, halt after reset using DBGRQ

Вывод RESET используется для сброса процессора. После сброса J-link непрерывно пытается остановить процессор. Это обычно останавливает CPU вскоре после сброса; процессор может в большинстве систем выполнить ряд некоторых инструкций, прежде чем он будет остановлен. Число выполненных инструкций зависит главным образом от скорости JTAG: чем быстрее скорость JTAG, тем быстрее CPU может быть остановлен. Некоторые процессоры могут быть остановлены перед выполнением инструкций, потому что имеется задержка после сброса

### LOG

Log файл может быть сгенерирован J-Link ARM и J-Link RDI. Этот файл может оказаться полезным для отладки и оценки работы. Он может помочь вам решить проблему самостоятельно. Или отослать его в службу поддержки.

### Использование удаленной отладки.

Удаленная отладка – весьма удобная вещь, в том случае, когда Вы находитесь далеко от Вашего отлаживаемого устройства. Этот тот случай, когда программист сидит в Москве, а устройство, которое надо отладить находится в Хабаровске. Итак, что для этого требуется.

- 1. Компьютер программиста с установленной средой IAR, например.
- 2. Internet/Ethernet доступ.
- 3. На компьютере программиста должны быть установлена Setup\_JlinkARM.

- 4. К удаленному компьютеру должен быть подключен USB JTAG адаптер.
- 5. На удаленном компьютере должны быть установлены драйвера к адаптеру и установлена Setup JlinkARM.
- 6. Программист должен знать реальный IP адрес удаленного компьютера. Для этого на удаленном компьютере необходимо проделать следующее:
  - Пуск -> Выполнить -> cmd

Запуск	ірограммы [?]
1	Введите имя программы, папки, документа или ресурса Интернета, и Windows откорет их
Открыт	
	ОК Отмена Обзор

• Набрать ipconfig и нажать Enter. На экране будет отображен IP адрес компьютера.

Запускаем IAR. Открываем проект, заходим в свойства проекта и на вкладе J-Link/J-Trace -> Connection выбираем TCP/IP и вводим адрес удаленного компьютера. Нажимаем кнопку Ok, чтобы закрыть окно Option.

На удаленном компьютере запускаем программу JlinkTCPIpserver.exe, которая обычно находится в папке Program File/Segger/JLinkARM\_Vxxx/

IPStat	Not connected		
	R	W	WR
This connection	0	0	0
Total	0	0	0
USBStat Not connected			
	📕 Stay on to	Р ,	About
Status	, cuy on to		4DOU(

Теперь в IAR можно смело нажимать Debug и работать.

Примечание: Большинство антивирусных программ воспринимает этот процесс отрицательно и блокирует его выполнение. Поэтому рекомендуется отключить их на время удаленной отладки.

### Использование программы JFlash.

Программа JFlash весьма удобна для массового программирования микроконтроллеров. Запускаем JFlash.

JetLink5 – USB to J7AG adapter

J-Flash ARM ile Edit View	V3.60a - [c:\Program Target Options Window	Files\SEGGER\JLink Help	ARM_V360a\Defaul	t.jflash *]		-	
Project - De	efault 💶 🗖 🔀						
Name	Value						
Connection	USB (Device 0)						
Sonnoodon	000 [00100 0]						
nit JTAG speed	10 kHz						
ITAG speed	Auto recognition						
AP number	<not used=""></not>						
- len	<not used=""></not>						
PU	ABM7/ABM9						
ndian	Little						
heck core Id	No						
ore Id	0x0						
se target RAM	No						
AM address	0x0						
AM size	4 KB						
	CEL a secolizati	C					
ash memory	0v0						
)rganization	16 bits x 1 chip		///seg	GER			
LOG							
plication log start J-Flash ARM V3.6 JLinkARM.dll V3. eading flash devic eading MCU devi eading MCU devic eading new project New project crea	ted 30a (J-Flash compiled Feb 9 30a (DLL compiled Feb 9 20 ce list (c:\Program Files\SEG es read successfully (139 De ce ist (c:\Program Files\SEG ces read successfully (134 D tel lie (c:\Program Files\SEG) ted successfully	2007 20:05:26) 107 20:05:02) GER V,LinkARM_V360a\E GER V,LinkARM_V360a\E evices) SER V,LinkARM_V360a\D	:TC\JFlash\Flash.csv] :TC\JFlash\MCU.csv] efault;flash]				
1							>

Далее, File -> Open Project, выбираем проект соответствующий Вашему процессору.

Теперь File - > Open, выбираем файл который надо записать. И наконец Target - > Auto , для стирания, программирования и проверки.

Примечание: Скорее всего, в данный момент Вы увидите окно с просьбой ввести номер лицензии. Подробнее о лицензии смотрите в пункте «Лицензии».

#### Лицензии.

Использование RDI и JFlash требует лицензии, которая продается отдельно фирмой Segger. С ценами можно ознакомится здесь http://www.segger.com/pricelist\_jlink.html

Однако, в Интернете можно найти файл под названием rdikeygen, который по серийному номеру адаптера генерирует все необходимые лицензии.

		×
J-Link seial	01010101010	ОК
RDI	License_RDI_V0_51010101010_E0_K2438c2eb	Generate
FlashBP	License_FlashBP_V0_51010101010_E0_Ke4e8ef77	
FlashDownload	License_FlashDL_V0_51010101010_E0_K20443577	
J-Flash	License_JFlash_V0_S1010101010_E0_Kfac2b54a	

Разумеется, пользоваться им можно ТОЛЬКО в ознакомительных целях.

#### **JTAG** Разъем.

Имеется две версии разъемов – 14 и 20 контактов. Назначение контактов можно увидеть ниже в таблице.

14 – контактная версия адаптера.

1	VTref	Вход	Питание от отлаживаемой платы
2	5V	Выход	Опция – вывод 5V из устройства.
3	nTRST	Выход	Сигнал сброса ЈТАС отлаживаемого контроллера
4	GND		
5	TDI	Выход	Выход ЈТАС данных адаптера
6	GND		
7	TMS	Выход	Выход сигнала установки режима ЈТАС
8	RTCK	Вход	Не используется
9	ТСК	Выход	Выход сигнала тактирования ЈТАС
10	GND		
11	TDO	Вход	Вход ЈТАС данных от отлаживаемого контроллера
12	RESET	Вх/Вых.	Сигнал сброса отлаживаемого контроллера
13	GND		
14	5V	Выход	Для совместимости с моделями KS

#### 20 – контактная версия адаптера.

1	VTref	Вход	Питание от отлаживаемой платы
2	Vsupply		
3	nTRST	Выход	Сигнал сброса ЈТАС отлаживаемого контроллера
4	GND		Земля
5	TDI	Выход	Выход ЈТАС данных адаптера
6	GND		Земля
7	TMS	Выход	Выход сигнала установки режима ЈТАС
8	GND		Земля
9	ТСК	Выход	Выход сигнала тактирования ЈТАС
10	GND		Земля
11	RTCK	Вход	Не используется
12	GND		Земля
13	TDO	Вход	Вход ЈТАС данных от отлаживаемого контроллера
14	GND		Земля
15	RESET	Вх/Вых.	Сигнал сброса отлаживаемого контроллера
16	GND		Земля
17	DBGRQ		Не используется
18	GND		Земля
19	DBGACK		Не используется
20	GND		Земля

JetLink5 – USB to J7AG adapter

### Обновление программного обеспечения (firmware).

Обновление прошивки производится автоматически, при запуске любой из программ, использующих библиотеку JLink.dll. Данная библиотека входит в состав Setup\_JlinkARM, который доступен по адресу - <u>http://www.segger.com/download\_jlink.html</u>