AN1215



# VORAGO VA108x0 GCC IDE application note

April 2, 2019 Version 1.1

#### VA10800/VA10820

#### Abstract

ARM has provided support for the GCC (GNU C compiler) and GDB (GNU DeBug) tools such that it is now a very reliable and often used development environment for ARM Cortex-M based MCUs. This application note covers all the steps necessary to get the suite running specifically with the VA108x0 MCU from VORAGO. This includes:

- Instructions on how to install the Eclipse+JRE (Java runtime environment), GCC and J-Link GDB server packages,
- Information on unique linker (\*.LD), make (Makefile) and startup (\*.S) files
- Example project demonstrations

During the development, several challenges were encountered with download, options and tool nuances. These are addressed in the frequently asked questions. It is highly recommended to review this list if you have plans to "open-the-hood" and modify linkers, make files, startup or other components in the environment.

## **Table of Contents**

1	Overview of GCC and GDB development environment	1
	Installing GCC and conducting a test run	
3	Installing Eclipse CDT	5
	Installing Segger J-LINK GDB server	
5	Downloading an Example VA108x0 project	7
6	Introduction to Eclipse	7
	Unique files for VA108x0 – make, startup and linker files	
8	Build and debug first project	.11
	Build and debug the full REB1 BSP project	
10	Creating your own project	.16
11	Installing OpenOCD (On-chip Debugger)	.16
12	Alternative probe - Black Magic Probe (BMP)	.18
13	Programming SPI NVM for the VA108x0	.20
14	Conclusions	.20
15	Common questions and issues	.20
	Other Resources	.21

# 1 Overview of GCC and GDB development environment

Eclipse is an integrated development environment (IDE) that has special support for C programming. The compiler and debugger interface are specified inside the project. Eclipse



plug-ins are used to access some of the debugger functions. Figure 1 shows a high-level interconnect diagram of all the pieces used. The following sections provide more depth into each block.

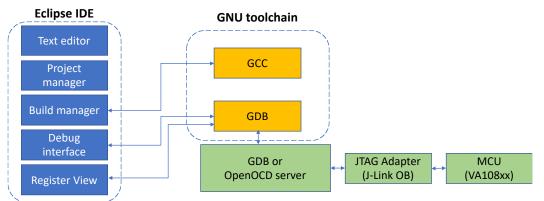


Figure 1- Development environment interconnect diagram

#### 1.1 GCC

The GNU Compiler Collection includes front ends for <u>C</u>, <u>C++</u>, Objective-C, <u>Fortran</u>, Ada, and Go, as well as libraries for these languages (libstdc++,...). The home URL is:

#### https://gcc.gnu.org

There is extensive documentation available for GCC on-line at: <u>https://gcc.gnu.org/onlinedocs/</u>. This information can sometimes come in very handy but by following the steps listed in this application note, the need to read the GCC documentation should be limited.

ARM has invested time and effort in improving the compiler specific to the ARM architectures. The ARM version of GCC has been proven on many MCUs and improved over the last 15 years. Information on this development and access to the source code can be found at:

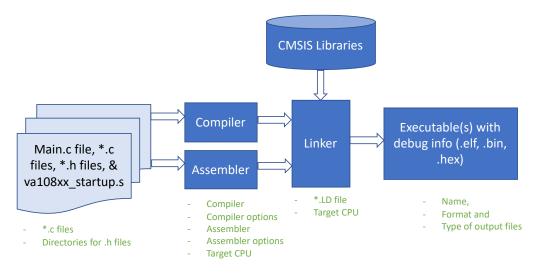
#### https://developer.arm.com/open-source/gnu-toolchain/gnu-rm.

This application note will use "arm-none-eabi-gcc". This is an embedded application binary interface (eabi) for the "arm" instruction set with no ("none") IC vendor specific information. The project specific linker and "Makefile" files specify which ARM core is used.

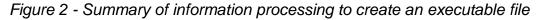
Figure 2 provides a summary of the flow and processing of information to create an executable file that the debugger can utilize. ARM has created a software interface standard known as "CMSIS", Cortex-M Software Interface Standard. This has multiple layers and a wide breadth but in general, it offers an API to lower level drivers for peripherals like UART, SPI and I2C. This is intended to make code very portable from one ARM based processor to another.



#### AN1215 - VA108x0 GNU GCC Application Note



"Makefile" (A text file inside the project) calls out all of the above information in green. The Linker uses a \*.LD file to receive: the ARM core, the memory map placement for instructions/data and size of stack.



#### 1.2 GDB - GNU Project Debugger

GDB is intended to provide a powerful user interface for debugging software on the target MCU. It includes the following capabilities:

- Download executable files to either RAM or NVM on the target MCU
- Start program from any program counter location
- Stop program execution by the use of break points or when a STOP instruction is executed
- Examine and modify memory mapped register and memory contents
- Examine and modify CPU registers (not memory mapped)

More information can be found at: <u>https://www.gnu.org/software/gdb/</u>

In order to support different JTAG / SWI probes, a GDB client server concept has been implemented. When the "debug" command is entered in Eclipse, the GDB client is launched. For our case, the client will be "arm-none-eabi-gdb". A port number is specified in the debug configuration Debugger menu. A PC will direct the communication traffic to a separate task controlling the probe. For the Segger J-Link, the port ID is 2331. The OpenOCD port ID is 3333. The GDB server can be started via a script from Eclipse or separately prior to entering the debug command.

As a side note, the fish sketch often accompanying GDB information is in remembrance of Fred Fish (November 4, 1952 – April 20, 2007) and his significant contributions to the GDB effort.



AN1215 - VA108x0 GNU GCC Application Note

#### GDB: The GNU Project Debugger

[bugs] [GDB Maintainers] [contributing] [current git] [documentation] [download] [home] [irc] [links] [mailing lists] [news] [schedule] [song] [wiki]

GDB: The GNU Project Debugger Figure 3 – GDB home page with tribute to Fred Fish.

# 2 Installing GCC and conducting a test run

ARM has taken the basic GCC suite and tailored it to the ARM architectures. Access to the latest version can be found at:

https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads

The name of the .exe file should look something similar to "gcc-arm-none-eabi-X\_X-XXXXX... .exe". See below for screen capture of the download site. Newer versions may be available when you visit the site.

Home / Linux and Open Source / CPNU Tooldan / CPNU Am Embedded Tooldaan / D GNU Arm Embedded Toola		
Overview Downloads		
Downloads The GNU Embedded Toolchain for Arm is a ready-to-use, open source tools for C, C++ and Assembly programming targeting Arm Cortex-M family of processors. It includes the GNU Compiler (GCC) and is avail charge directly from Arm for embedded software development on Wi and Mac OS X operating systems. Follow the links on this page to dow right version for your development environment.	and Cortex-R ible free of ndows, Linux	GNU Arm Embedded Toolchain Version 7-2017-q4- major Released: December 18, 2017
GNU Arm Embedded Toolchain: 7-2017-q4-major December 18, 2017		
What's new in 7-2017-q4-major In this release	Windows 32-bit File: gcc-arm-none-eabl-7-20 n32.exe (82.53 MB) Windows 32-bit File: gcc-arm-none-eabl-7-20	
gcc-arm-none-eabl-7-2017-q4-major-win32.exe Windows 32-bit Installer MD5: bb4def39ff1cb3ff5d2931597d9aea4e	n32-sha1.exe (82.53 MB) Windows 32-bit File: gcc-arm-none-eabi-7-201 n32-sha2.exe (82.53 MB)	
2 gcc-arm-none-eabi-7-2017-04-major-win32-sha1.exe	Windows ZIP File: gcc-arm-none-eabl-7-201 p22 zin (122 71 MP)	17-q4-major-wi Download

Figure 4 - ARM GNU download site screen capture

Double click on the \*.exe file and click through the install instructions. Make certain that you click the "Add path to environment variable". In order for Eclipse to access the GCC software, the global path must be changed to include the folder containing GCC executables.



#### 2.1 GCC test run

Open a command prompt (CMD) window and enter the following command to show the GCC version information and confirm it was installed: "arm-none-eabi-gcc -v". This command will show the version information. The below figure shows an example screen capture.

🚾 Command Prompt	-		
microsoft Windows [Version 10.0.16293.371] (c) 2017 Microsoft Corporation. All rights reserved.			
:\Users\demol>arm-none-eabi-gcc -0			
sing built in specs.			
OLLECT_GCC=arm-none-eabi-gcc OLLECT_LTO WRAPPER=c:/program\ files\ (x86)/gnu\ tools\ arm\ embedded/7\ 2017-g4-major/bin//lib/gcc/arm-none-eabi/7.2.1/lto-wr			
orrer_rrs_waterrever,program( rres( (xbo)/gnu( coors( arm( embedded)/( zer/qe-major/br///rr/gc/arm/none-eabr///r///	apper sexe		
onfigured with: /tmp/jenkins/jenkins-GCC-7-build toolchain docker-633 20171130 1512067137/src/gcc/configurebuild=x86 64-linux	enuhor	st=1686-	w64-
ngw32target=arm-none-eabiprefix=/tmp/jenkins/jenkins-GCC-7-build toolchain docker-633 20171130 1512067137/install-mingw			
/jenkins-GCC-7-build_toolchain_docker-633_20171130_1512067137/install-mingw/libinfodir=/tmp/jenkins/jenkins-GCC-7-build_toolc			
a_1512067137/install-mingw/share/doc/gcc-arm-none-eabi/infomandir=/tmp/jenkins/jenkins-GCC-7-build_toolchain_docker-633_20171			
ningw/share/doc/gcc-arm-none-eabi/manhtmldir=/tmp/jenkins/jenkins-GCC-7-build_toolchain_docker-633_20171130_1512067137/instal			
rm-none-eabi/htmlpdfdir=/tmp/jenkins/jenkins-GCC-7-build_toolchain_docker-633_20171130_1512067137/install-mingw/share/doc/gcc able-languages=c.c++enable-mingw-wildcarddisable-decimal-floatdisable-libffidisable-libgompdisable-libmudflapd			
able-languagessc,c++enable-mingw-wildCarddisable-dechmai-tloatdisable-libttidisable-libgompdisable-libmudtlapd sable-libsspdisable-libstCxx-pchdisable-nlsdisable-shareddisable-threadsdisable-tibwith-gru-as			
abue-riosspusable-ribarocxypthdisable-risable-risable-ribarock-ribarock-ribarock-ribarock-risable-risable-ribarock-ribaro			
ngw/arm-none-eabiwith-libiconu-prefix=/tmp/jenkins/jenkins-GCC-7-build toolchain docker-633 20171130 1512067137/build-mingw/arm			
/tmp/jenkins/jenkins-GCC-7-build toolchain docker-633 20171130 1512067137/build-mingw/host-libs/usrwith-mpfr=/tmp/jenkins/je			
hain_docker-633_20171130_1512067137/build-mingw/host-libs/usrwith-mpc=/tmp/jenkins/jenkins-GCC-7-build_toolchain_docker-633_2	9171130_1	51206713	7/bi
d-mingw/host-libs/usrwith-isl=/tmp/jenkins/jenkins-GCC-7-build_toolchain_docker-633_20171130_1512067137/build-mingw/host-libs			
/jenkins/jenkins-GCC-7-build_toolchain_docker-633_20171130_1512067137/build-mingw/host-libs/usrwith-host-libstdcxx='-static-l	ibgcc -Wl,	,-Estati	
dc++,-Bdynamic -lm'with-pkgyersion='GNU Tools for Arm Embedded Processors 7-2017-q4-major'with-multilib-list=rmprofile			
hread model: single inc. wersion 7,2.1 20120004 (release) [ARM/embedded-7-branch rewision 255204] (GNU Tools for Arm Embedded Processors 7-2017-04-mai			

*Figure 5 - Command prompt invoking GCC and showing the version information.* The GDB client software was also installed with the GCC download. You can confirm this by entering: "arm-none-eabi-gdb -v" at a command prompt.

# 3 Installing Eclipse CDT

Eclipse is a free IDE (integrated debug environment) that supports several different programming languages. For our purpose, we will focus on the CDT (C/C++ Development tooling) version. Please visit the below URL for access to the latest version.

#### https://www.eclipse.org/downloads/

As of April 2, 2019, the download filename was: eclipse-inst-win64.exe.



## AN1215 - VA108x0 GCC IDE Application Note

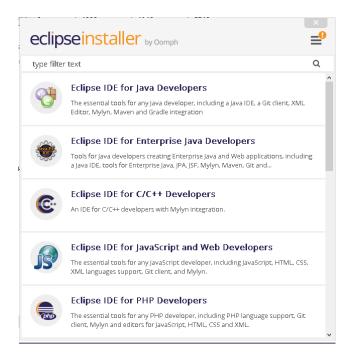


Figure 6 – Eclipse install screen capture

Download Eclipse for your platform. Run the executable file and select "Eclipse IDE for C/C++ Developers" as shown in the Figure 6. You will want to remember the folder location where you extracted the files. We used Windows 64-bit and placed them in a folder called "c:/EclipseCDT" but any folder name should work.

Eclipse requires Java runtime environment, so you will have to also download JRE from <u>http://java.com</u>. After downloading the "jre-8u171-windows-x64.exe" run the program and click the "change destination" box. Before starting the install, create a folder "jre" in the "eclipse" folder created in the above paragraph. Change the install location to the "jre" folder just created.

Alternatively, it is possible to modify the eclipse.ini file to point to the JRE path location. In that case, use the default location of c:\ProgramFiles\Java\jre1.8.0\_171.

Detailed information on CDT can be found at: https://wiki.eclipse.org/CDT/User/NewIn90#Release

# 4 Installing Segger J-LINK GDB server

Since the REB-1 development board comes with J-LINK OB, we are using the J-LINK GDB server to route information from the GDB tool to the MCU. The JLINK GDB server is part of the larger J-LINK download available at: <u>https://www.segger.com/jlink-software.html</u>



Select Software and documentation pack for Windows V6.10n [23,162 KB]. If a later version is available, please use it.



Figure 7 - Segger J-Link download site screen capture

# 5 Downloading an Example VA108x0 project

Two projects are available on GitHub: Blink\_only and the full REB1 BSP. They can be found at: <u>https://github.com/Voragotech</u>

Download both projects and put them in the same directory. For our example, we used: c://Vorago\_projects. The next section will access these projects through Eclipse.

•E 🗢 Help	o - Eclipse Platfor	m () \	/oragotech/	VA108x0_G ×	+ ~								-	
$\rightarrow$ O	<u>م</u>	GitHub, Inc.	[US] https;//	/github.com/Vorag	gotech/VA10	08x0_GCC					${\simeq}$	饽	h	Ŀ
Search o	er jump to		/	Pull requests	Issues	Marketpla	ice Explo	ore				4	• +	۰E
Voragote	ch <b>/ VA108x</b>	_GCC							⊙ Watch •	0	★ Sta	ar O	¥ For	k 0
<> Code	() Issues 0	ነን Pull requ	ests 0	III Projects 0	🗏 Wik	ci 📊 Ir	nsights	() Se	ttings					
fanage topics	on and examp	le GCC proje		1 branch		0	0 release	24			#1	contribu	tor	Edit
fanage topics	) 9 commits			1 branch		2	0 release		Upload fi	les Fi	匙 1 nd File		tor or down	Edit
fanage topics	9 commits	quest	ν	1 branch		٢			Upload fi		nd File		or down	oad <del>-</del>
fanage topics		quest	ν	1 branch Blink only p	roject for R		Create ne		Upload fi		nd File	Clone o ommit Øee	or down	load ▼ t now
Branch: master	• 9 commits • New pull re e Full REB1 BSP pr linkonly.7z	quest	ν		,	EB1 - GCC	Create ne		Upload fi		nd File	Clone o ommit Øee	or down 7da0 jus minute	load 🗸
Annage topics (7) Branch: master · (1) rbannatyne (2) Prj_REB1_b	O 9 commits     New pull re     full REB1 BSP pn linkonly.7z EB1_BSP.7z	quest	ν	Blink only p	SP project in	EB1 - GCC	Create ne		Upload fi		nd File	Clone o ommit Øee 2	or down 7da0 jus minute	oad - t now s ago t now

Figure 8 - VORAGO github page

# 6 Introduction to Eclipse

The organization of a project is somewhat unique in Eclipse and deserves a quick explanation. Eclipse uses a "workspace" concept which groups together the following in a folder:



- 1. Some configuration pertaining to all these projects in metadata
- 2. Some settings for Eclipse itself
- 3. Optionally one or more related projects

This happens by creating a directory/folder and storing files in it that provides Eclipse this information when the workspace is opened or "switched" to. See below for the contents of an example Workspace folder.

📙   🛃 📙 ╤   C:\VA108∞_GCC	C\Wksp_VOR_RI	EB1				_	
File Home Share V	/iew						$\sim$
$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\blacksquare$ $\Rightarrow$ This PC	> Windows7_0	OS (C:) → VA108∞_GCC → Wksp_VOI	R_REB1		~ 0	Search W	′ksp_V0 🔎
📌 Quick access	<b>^</b>	Name	Date modified	Туре		Size	
Desktop	*	.metadata	6/7/2018 3:51 PM	File folder			
		Prj_VOR_REB1	6/4/2018 10:16 AM	File folder			
Downloads SiliconSpaceTechnology	*	RemoteSystemsTempFiles	3/11/2018 9:50 AM	File folder			

Figure 9 - Contents of Workspace directory

Copying a workspace from one machine to another can be dangerous as the metadata can reference information that is not available on the second machine. Instead it is recommended that a new workspace be created and then reference the project.

Eclipse makes it easy to create a new workspace and to import an existing project. With Eclipsed already launched, Use the "File" pull down menu to select "Switch Workspace". You can either point to an established workspace or to an empty folder.

With Eclipse already launched and a Workspace active, pull down the following menu: File -> New -> Makefile Project with Existing Code. You should get a screen similar to Figure 10 below. Unclick the C++ box and select "Cross GCC".



#### AN1215 – VA108x0 GCC IDE Application Note

New Project	0		×
Import Existing Code			
Create a new Makefile project from existing code in that same directory			
Project Name			
Prj_REB1_blinkonly			
Existing Code Location			
C:\software_reb_gcc\eclipse_2018\projects\Prj_REB1_blinkonly		Brows	e
Languages ☑ C □ C++			
Toolchain for Indexer Settings <none></none>			
Cross GCC GNU Autotools Toolchain			
Show only available toolchains that support this platform			
? Finish	(	Cancel	

Figure 10 - Eclipse New -> Makefile Project with Existing Code screen capture

The workspace is now associated with the referenced project and you can start your code development from a solid starting point.

#### 6.1 Eclipse first time use

Open Eclipse by double clicking on the eclipse.exe file that was downloaded per instructions in Chapter 4. After a few seconds a menu will appear asking for a workspace. Point to ../VA108X0\_GCC\_project/Workspace-VORAGO-REB1-blinkonly folder downloaded in section 5. Note that once Eclipse has been used, it will remember the last used Workspace and will go directly to it. If a prompt for a Workspace is not shown, use the "file" pulldown menu and select "switch workspace", then proceed to the above-mentioned workspace.

You should see something similar to the below screen capture.

<ul> <li>C/C++ - vorago_blink_only/src/main.c - Eclip</li> <li>File Edit Source Refactor Navigate Searc</li> <li>Image: Image: Im</li></ul>		• = -	- 다 ×
Project Explorer     □       □     ♀       ↓     ∅        ↓     ∅   <	<pre>Makefile @ mainc %% @ startup_val08xxS @ val08x_linker.id 39 */ 40@ int main (void) 41 ( 42 valatile int count = 1 ; /* counts the # of blinks */ 43 44 /* Enable clock for all peripherals */ 45 VOR_STSCONFIG-&gt;PERIPHERAL_CLK_ENABLE_UARTS   CLK_ENABLE_PORTA   0 46 CLK_ENABLE_UARTA   CLK_ENABLE_UTILITY   CLK_ENABLE_1 48 49 50 51 /* Configure PA[7]. This GPIO vill turn ON LED D3. */ 52 VOR_GPIO-&gt;BANK[0].DIR  = (1 &lt;&lt; PORTA_7_03); 53 /* Configure PA[7]. This GPIO vill turn ON LED D3. */ 54 VOR_GPIO-&gt;BANK[0].DIR  = (1 &lt;&lt; PORTA_7_03); 55 VOR_GPIO-SANK[0].DIR  = (1 &lt;&lt; PORTA_7_03); 55 VOR_GPIO-SANK[0].PIR  = (1 &lt;&lt; POR</pre>	CAI	Coutine CS Make Tar Task List  Coutine CS Make Tar  Mark List  Coutine CS Make Tar  Mark List  Coutine Tar  Mark List  Coutine Tar  Mark List  Mark L
📄 REB1_blink_rom.hex 📄 REB1_blink.map 📄 REB1x_blink.map	🖹 Problems 🧔 Tasks 😡 Console 🛛 🔲 Properties No consoles to displey at this time.		≝ ≅ ≠ ≅ ≠ ₽ □ ■E <b>№R №</b> ₽ □ ■ <b>≭ %</b> ₿

Figure 11 - Blink\_only project screen capture



#### 6.2 Updates and Installing New Software

The native download of Eclipse does not support the "GDB Hardware Debugger" Configuration and an extra step is necessary to enable GDB use.

To make certain Eclipse is using the latest tools for C development it is recommended that you pull in the latest updates. This is done using the following pull-down menu: Help -> Check for update.

For some versions of Eclipse such as Mars, it might be necessary to use the install new software option instead of the update option. In that case, install updates for CDT. This will enable the creation of new GDB debug configurations.

## 7 Unique files for VA108x0 – make, startup and linker files

Inside each project are three text files that determines which files are included in the project, how the MCU boots and which memory space the program should use. Each file can be viewed and modified in the eclipse environment using the project explorer. The following three sections describe these files.

#### 7.1 Calling out project options in "Makefile"

The "makefile" is a text file normally >200 lines long that contains the files and options needed to build a project. Included in this list are:

- Compiler to be used
- Target device CPU information. i.e. cortex-m0.
- Linker file to be used
- List of \*.c files to include
- Directories to find the \*.h files.
- Output file name and list of formats i.e. .hex, .bin, .elf.

We have provided a makefile for the two example projects. It resides in the root directory of the project. If you decide to add source files or add directories for .h files, you will probably need to modify this file.

# 7.2 Defining reset / interrupt vectors and initializing MCU before main{} is called – "startup\_VA108x0.S"

Explicit definitions for the reset and interrupt vectors are called out in this file. The Cortex-M0 has 16 core related vectors and 32 User interrupt vectors. IRQ[0:31]. VORAGO has designated the first 12 interrupt vectors to be specific to a specific peripheral on the device



such as SPI0. This is not a convention that must be followed but for ease in understanding these were named with the peripheral assigned to the vector.

The reset handler and the default interrupt handler are also included in the startup\_VA108x0.s file. This is a text file with assembly language instructions and directives. Prior to the main.c code executing structure information is moved from ROM space to RAM space. On some MCUs it is necessary to have some parameters setup and a system\_init routine called. Commonly the clock must be configured as the first step. The VA108x0 device has a straightforward clock system that does not need to be setup and the reset handler moves a section of ROM data to RAM, initializes the first 4 registers then jumps to the main routine.

#### 7.3 VA108x0\_linker.ld

The linker file provides information that allows the build process to attach object file information to specific addresses in the MCU. The linker file is a text file that can be easily modified. It contains directives for certain pieces of information to be grouped together such as data and text. The linker file has labels that are used by the startup.s file to move information from ROM to RAM.

For first time GCC users, it is recommended that the linker file be left untouched. After the environment is better understood, changes to the linker file can be attempted.

# 8 Build and debug first project

As with the time-honored tradition of many other MCU evaluation kits the first project will be a simple LED blink routine. The project is located in the folder of the downloaded Prj\_REB1\_blinkonly.7z file. Follow the steps in the next three sections to launch, build and execute the project.

#### 8.1 Start Eclipse and load the project

Find the directory with Eclipse and double click on the eclipse.exe file. After a few moments, Eclipse should open and ask for a "workspace". Using Windows File Explorer create an empty folder and title it "Wksp\_VOR\_Blink" or any similar title. Navigate to the directory and hit the OK button. At this point, the project files are not associated with the workspace. Under the eclipse "file" menu, select "new" and then "Makefile project with Existing Code". Set the existing code location to point to the directory with material from the download in section 5, "Prj\_REB1\_blinkonly". Unclick the "C++" box and select "Cross GCC" as the toolchain. Then hit finish.



🖨 Workspace Launcher		Select Workspace Directory	×
<b>Select a workspace</b> Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.		Select the workspace directory to use.	<b>^</b>
Workspace: C:\\/A108x_GCC\\Wksp_VOR_REB1	V Browse	<ul> <li>VA108x_GCC</li> <li>Wksp_VOR_Blink</li> <li>metadata</li> <li>RemoteSystemsTempFiles</li> <li>vorago_blink_only</li> <li>Wing_VOB_DETA</li> </ul>	~
	OK Cancel	K     Folder: Wksp_VOR_Blink     Make New Folder OK Cance	>

Figure 12 - Eclipse Workspace Launch screen capture

#### 8.2 Build the project

The "blink\_only" project has a build configuration file. It is possible to either use the pull-down menu (Project -> Build All) or click on the build icon (Hammer shape) as shown below. Different versions of Eclipse have slightly different appearance and this capture may not be identical to yours.

C/C++-Eclipse File Edit Source Refactor Navigate Searc	h Project Run Window Help			-		×	2
1 🖆 • 🗟 🖄 🖆 👋 • 🖓 • 🕅 1 🗙 1 🖄	• 18 • 12 • 17 • 14 • 10 • 9, • 9, • 14 • 11 12 10 10 10 10 10 + 10 • 10 • 10	•	Quick	Access	<b>B</b>	<b>† R</b>	
Project Explorer 22	Malefie      mainc 23      tatup_va108o.5		E Outline 🕄 🛞 Make Tar. E Jag	. 🗐 Ты ९ 📌 Ф		 	
(c)         (c)         (c)           (c)         (c)         (c)	<pre>valatile int count = 1 ; /* counts the # of blinks */ /* Exable clock for all peripherals */ vog_strockTrio-repripherals // clk_Exable_cost / clk_Exable_cost / clk_Exable_cost // clk_Exable_cost //</pre>	•	<ul> <li>Valuac.n</li> <li>Vinsec.im, timeout: 1</li> <li>PORTA_10_D2</li> <li>PORTA_10_D2</li> <li>PORTA_604</li> <li>PORTA_11_SM_USER</li> <li>main(void): int</li> <li>SysTick_Handler(void)</li> </ul>				
REB1_blink_rom.hex REB1_blink.map	👔 Problems 🧔 Tasks 📴 Console 💠 🔲 Properties		d 🛛 - 🖬 - 🗆 🕴	E 🖻 R	23	- 0	1
REB1x_blink.map	No consoles to display at this time.				×ÿ	創事	

Figure 13 - Blink\_only project "C" window capture

#### 8.3 Download and run the program

The "blink\_only" project may not have debug configuration file. It takes three quick steps to setup a configuration as shown in the following three figures. Start by selecting "Debug Configuration" under the "Run" menu. In the left-hand menu select "GDB Hardware Debug" and then the "New" icon. Follow the instructions in the next three sections.

#### 8.3.1 Select Project and Application.

Hit the "Browse" button to the right of the project field. It should show the active project immediately. Then hit the "Search Project" button by the application field.



#### AN1215 - VA108x0 GCC IDE Application Note

Debug Configurations					$\times$
Create, manage, and run co	onfigurations			X	5.
Image: Construction     Image: Construction       Imag	Name: Segger, GOB, debug Main Segger, GOB, debug Project: Prj.REB1, blink only C/C ++ Application: REB1x_blink_rom.elf Build (df required) before launchin Build Configuration: Use Active © Enable auto build © Use workspace settings	Variables g O Disabl	ommon)		
< >> Filter matched 10 of 10 items			Revert	Apply	
?			Debug	Close	e

Figure 14 - Screen 1 of debugger setup

#### 8.3.2 Set hardware

Click the "Debugger" tab and enter information as shown. The "arm-none-eabi-gdb" will invoke the ARM specific GDB interface software. Port 2331 is the Segger GDB server.

Debug Configurations			
Create, manage, and run co	nfigurations		Ŕ
Image: Solution of the second sec	Name: Segger_GDB.debug Main St Debugger Startup Source Cor GDB Setup GDB Command: arm-none-eabi-gdb Remote Target Use remote target Current timeout (seconds): JTAG Device: Generic TCP//P Host name or IP address: localhost Pot number: 2331 Force thread list update on suspend	mmon Browse	Variables
< >> Filter matched 10 of 10 items		Revert	Apply
(?)		Debug	Close

Figure 15 - Screen 2 of debugger setup

#### 8.3.3 Set Startup parameters

Later versions of the eclipse IDE do not support the Reset & Delay button. It is necessary to manually enter the two initialization commands for the Segger GDB server. If this is not done, the debugger will attempt to run code without resetting the MCU. This can lead to unpredicted



results with peripherals still running from the last processor run. Hit the "Apply" button to save this configuration.

🗅 🖻 🎨 🗎 🗶 📄	Name: Segger_GDB_debug
type filter text C //C++ Application C //C++ Attach to App C //C++ Attach to App C //C++ Postmortem I C //C++ Postmortem I C //C++ Remote Appli C //C++ Attach Appli C /	Main ( Debugger ) Startup Source Common Initialization Commands Reset and Delay (seconds)): Hait monitor reset monitor reset monitor hait
C New_configuration Launch Group Launch Group (Depre	Load Image and Symbols  Load image  Use project binay: REB1x_blink_rom.elf  Use file:  Image offset (hex):  Load ymbols
	Ouse grintesinger, REB1x_blink_rom.elf     Ouse file: Workspace File System      Symbols offset (hed):      Due Command:
< >> Filter matched 10 of 10 items	Revert Apply

Figure 16 - Screen 3 of debugger setup

Before invoking the debugger (by hitting the "Debug" button), please have the REB1 board attached to the PC USB port and start the Segger JLink GDB server by navigating to the Windows Explorer -> Segger ->J-Link GDB Server V6.30i. Enter "Cortex-M0" if prompted for a target device and hit the OK button.

Chebug 32     Company blink, only, LINE (508     Control of Sprail 201784)     Control of Sprail 201784/Trace/breakpoint trap     Company blink only and the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap     Company blink on the structure of Sprail 201784/Trace/breakpoint trap	() 🕫 🖻 🖬 🖻
	eral Purpose and FPU Registi
	\$ B C C
67         00000158: cmm r1, #0           00000158: beq.n 0x154 cmm1+62>         00000158: beq.n 0x154 cmm1+62>           71         count ++ ;	
70         (           71         count ++ j           72         meec_tim_timeout = 0 j           73         VOR_OPIO=>NANK[0].TOGOUT  = (1 << FORTA_1	
75 ) 73 (VO_GPIO->BANK[0].TOO 76 (VO_GPIO->BANK[0].TOO 77 (VO_GPIO->BANK[0].TOO 00001565 orra_tl_t2	HOUT  = (1 << PORTA
	>
Console 🖄 🔄 Tasks 👔 Problems 🕠 Executables 🚺 Memory Browser 🗍 Memory 🖾 Console 🛛 🗮 💥 🙀 🚉 🐼 🖾 🐼 💭 💭 🚱 ago Julink, only "LINK, GOB (GoB Hardware Debugging) arm-none-eabi-gdb (8.0.58.20171128)	iii   ≝ ⊑ • ⊡ • °

Figure 17 - Invoking the debugger screen capture



Debug - Source not found Eclipse				- 🗆 ×
File Edit Navigate Search Project Pydev Run Window Help				
Image: A state of the state	* 🎽 🎘 🎯 🔅 🕶 🔾 🕶 💁	• 🕞 🖉 • 👌 • 🖗 • 🗭 • 🔶 •	⇒ *	Quick Access 📑 👘
Na Debug 23	🍇 🕅 🖃 🖬 🗢 🗖	(x)= Variables 🐁 Breakpoints 🍕 Ex	pressions 🔢 Registers 🕄 🛋 Mod	ules 🗖 🗖
C vorago_blink_only_LINK_GDB (GDB Hardware Debugging)     W    REB1x_blink_rom.elf		Name	Value	🚈 📲 📄 📑 😁 🗢 Description
P Thread #1 57005 (Running : User Request)			value	
🚚 am-none-eabi-gdb (8.0.50.20171128)		> 👫 General Registers		General Purpose and FPU Register
				ů.
		<		>
Makefile 🖻 main.c 🗟 startup_va108oc.S 💽 0x252a 😂	- 0	📴 Outline 🏧 Disassembly 😒	Enter location here 🗸 🧔	) 🗄 🔽 🔍 📑 😁 🔻 🗖
No source available for "0x252a"		No debug context		
				^
View Disassembly				

Figure 18 - Debug screen capture for Blink\_only project

Start the program execution by hitting the "resume" button as shown in the below figure. Alternatively you can hit "F8" or use the "Run" pull-down menu.

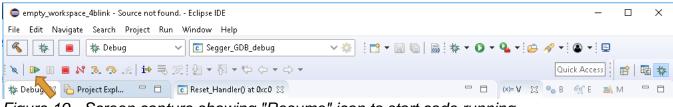


Figure 19 - Screen capture showing "Resume" icon to start code running

# 9 Build and debug the full REB1 BSP project

#### 9.1 Start Eclipse and load the project

Find the directory with Eclipse and double click on the eclipse.exe file. After a few moments, Eclipse should open and ask for a "workspace". Using Windows File Explorer create an empty folder and title it "Wksp\_REB1\_BSP" or any similar title. Navigate to the directory and hit the OK button. At this point, the project files are not associated with the workspace. Under the eclipse "file" menu, select "new" and then "Makefile project with Existing Code". Set the existing code location to point to the directory with material from the download in section 5, "Prj\_VOR\_REB1\_BSP". Unclick the "C++" box and select "Cross GCC" as the toolchain. Then hit finish.

If Eclipse was already started, use the "File -> Switch Workspace" menu instead of restarting Eclipse.

#### 9.2 Build the project

The "Prj\_VOR\_REB1\_BSP" project has a build configuration file, "default". This file calls out the GCC compiler and has library include paths. It is possible to either use the pull-down menu (Project -> Build All) or click on the build (hammer) icon to start the build process.



#### 9.3 Download and run the program

The "Prj\_VOR\_REB1\_BSP" project may not have a debug configuration file called "Prj\_VOR\_REB1". If not, please follow identical instructions as called out in section 8.3. This file calls out which file to download, the debugger hardware and startup options. Before launching the debugger, please have the REB1 board attached to the PC USB port and start the Segger JLink GDB server by navigating to the Windows Explorer -> Segger ->J-Link GDB Server V6.30i. Enter Cortex-M0 is prompted for a target device and hit the OK button.

## 10 Creating your own project

Instead of starting from scratch to create your project, it is recommended to start from the REB1 project. One way of doing this is to copy the entire folder Prj\_VOR\_REB\_BSP and renaming it. This provides a working file structure to start from.

# 11 Installing OpenOCD (On-chip Debugger)

Note: If you are not using an OpenOCD based setup, you can skip this section.

OpenOCD is standard to provide common debug and programming commands through an array of probes. OpenOCD can create a GDB server with port ID = 3333.

There is an organization that supports OpenOCD with a URL: <u>http://openocd.org</u>. The download location for the server software is: <u>https://sourceforge.net/projects/openocd/files/</u>

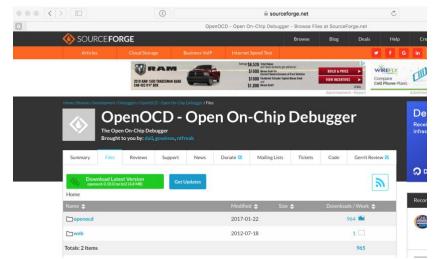


Figure 20 - OpenOCD download site screen capture



#### **11.1 Instructions for using the OpenOCD**

There is a plug-in for OpenOCD. See: <u>https://gnu-mcu-eclipse.github.io/debug/openocd/</u>. By following the steps, it is possible to create a debug configuration which will configure all the specifics of the probe, GDB server and OpenOCD.

#### 11.1.1 Adding OpenOCD to path environment

When GCC was loaded, there was an option to have the Environment path updated. For OpenOCD, this must be done manually. In Windows 10, open a Windows Explorer screen and right click on "This PC". Click on "Properties".

👱 Contr	ol Panel\System and Security\	System						_		×
$\leftarrow \rightarrow$	🕤 🛧 👱 > Control Panel	> System and Security > System	stem			~	ð	Search Cor	ntrol P	P
File Edi	it View Tools									_
Contr	rol Panel Home	View basic information	about your computer	System Properties	5					×
😌 Devic	e Manager	Windows edition		Computer Name	Hardware	Advanced :	System P	rotection	Remote	
	ote settings m protection	Windows 10 Pro © 2018 Microsoft Corporat	ion. All rights reserved.	You must be log Performance	ged on as ar	n Administrato	r to make	e most of th	ese chanç	jes.
🌳 Advar	nced system settings	System Processor:	AMD A6-5350M APU with Radeon(tm) HD Gr.	Visual effects,	processor sc	heduling, mer	noty usa	_	ual memor Settings	y
		Installed memory (RAM): System type: Pen and Touch:	8.00 GB (7.15 GB usable) 64-bit Operating System, x64-based processo Touch Support with 2 Touch Points	User Profiles	gs related to	your sign-in		:	Settings	
		Computer name, domain, and	workgroup settings	Startup and Re System startup		re, and debug	gging info	ormation		
		Computer name: Full computer name:	demo1-THINK demo1-THINK					:	Settings	
		Computer description: Workgroup:	WORKGROUP				0	Environme	nt Variable	s
		Windows activation	d the Microsoft Software License Terms			DK		Cancel	Ap	oply

Figure 21 - System variable change navigation procedure



#### AN1215 - VA108x0 GCC IDE Application Note

/ariable	Value	^
.M_LICENSE_FILE	c:\flexIm\LicensePlatinumUSB.dat;c:\flexIm\license.dat	
OneDrive	C:\Users\demo1\OneDrive	
Path	C:\Program Files (x86)\GNU Tools ARM Embedded\7 2017-q4-majo	
SNPSLMD_LICENSE_FILE	c:\flexIm\LicensePlatinumUSB.dat;C:/flexIm/license.dat	
SYNPLCTYP_LICENSE_FILE	c:\flexIm\LicensePlatinumUSB.dat;C:/flexIm/license.dat	
TEMP	C:\Users\demo1\AppData\Local\Temp	
ТМР	C:\Users\demo1\AppData\Local\Temp	$\checkmark$
	New Edit Delete	
stem variables		
	Value	^
Variable	Value C:\Program Files (x86)\AMD APP\	^
Variable AMDAPPSDKROOT		^
Variable AMDAPPSDKROOT BenchVueAnalyticsSetting	C:\Program Files (x86)\AMD APP\	^
Variable AMDAPPSDKROOT BenchVueAnalyticsSetting BenchVueInstallDir	C:\Program Files (x86)\AMD APP\ true	^
Variable AMDAPPSDKROOT BenchVueAnalyticsSetting BenchVueInstallDir CommandExpertBinPath	C:\Program Files (x80)\AMD APP\ true C:\Program Files (x80)\Keysight\BenchVue\	^
stem variables Variable AMDAPPSDKROOT BenchVueAnalyticsSetting BenchVueInstallDir CommandExpertBinPath COMMPath ComSpec	C:\Program Files (x80)\AMD APP\ true C:\Program Files (x80)\Keysight\BenchVue\ C:\Program Files (x80)\Keysight\Command Expert	^
Variable AMDAPPSDKROOT BenchVueAnalyticsSetting BenchVueInstallDir CommandExpertBinPath COMMPath	C:\Program Files (x80)\AMD APP\ true C:\Program Files (x80)\Keysight\BenchVue\ C:\Program Files (x80)\Keysight\Command Expert C:\Program Files\Lenovo\Communications Utility	*
Variable AMDAPPSDKROOT BenchVueAnalyticsSetting BenchVueInstallDir CommandExertBinPath COMMPath ComSpec	C:\Program Files (x80)\AMD APP\ true C:\Program Files (x80)\Keysight\BenchVue\ C:\Program Files (x80)\Keysight\Command Expert C:\Program Files\Lenovo\Communications Utility C:\WINDOWS\system32\cmd.exe	~

Figure 22 - System variable change -> path edit

Select the "Path" line and hit the "Edit" button. Add the OpenOCD binary directory which should look similar to ...\openocd-0.6.0\bin-x64".

# 12 Alternative probe - Black Magic Probe (BMP)

There are several low-cost (<\$100) JTAG probes that can be used with GDB. Information on one of the more popular ones, Black Magic Probe (BMP), can be found here: <a href="https://github.com/blacksphere/blackmagic/wiki">https://github.com/blacksphere/blackmagic/wiki</a>.

A unique debug configuration ("vorago\_blink\_with\_BMP") has been made for the Black Magic probe for the "blink\_only" file. Instead of using the "remote target" with port address, BMP uses a virtual communication port (USB CDC ACM). There is no need to start a GDB server prior to starting a debug section. The following three screen captures show the debugger setup menus.



## AN1215 – VA108x0 GCC IDE Application Note

Create, manage, and run configure	ations		1
- · · · · · · · · · · · · · · · · · · ·	Name: vorago_blink_with_BMP		15
type filter text	📄 Main 🔅 Debugger 🕨 Startup 🦆 Sour	ce 🔲 Common	
C/C++ Attach to Application C/C++ Postmortem Debugger C/C++ Remote Application C/C++ Unit D/C/C++ Unit	Project: vorago_blink_only C/C++ Application: C:\\4A1080_GCC\Wkxp_VOR_Blink\vorago_blin	k.onl/REB1.blink.rom.elf	Browse
GDB Hardware Debugging     C vorago_blink_only_LLINK_GDB     vorago_blink_with_BMP     fronPython Run     fronPython unittest	Build (if required) before launching Build configuration: Use Active	Variables Search Project.	Browse
a <sup>™</sup> Jython run a <sup>₩</sup> Jython unittest ▶ Launch Group	C Enable auto build     Use workspace settings	O Disable auto build Configure Workspace Settings	v

Figure 23 - Main debug configuration menu

Debug Configurations Create, manage, and run configu	rations
□ 🗎 🗶 📄 🔹 🕶	Name: vorago_blink_with_BMP
type filter text	📄 Main 🏇 Debugger 📦 Startup 🧤 Source 🔲 Common
C/C++ Attach to Application C/C++ Postmortem Debugger C/C++ Remote Application C/C+ Unit D/C/C++ Unit C/C++ Unit GDB Hardware Debugging C vorago_blink_only_ULINK_GDB	GDB Setup GDB Command: arm-none-eabi-gdb Remote Target Use remote target
© vorago_blink_with_BMP ≇ <sup>T</sup> IronPython Run ¥ <sup>J</sup> IronPython unittest ≇ <sup>J</sup> Jython run	JTAG Device: Generic TCP/IP  Host name or IP address: localhost
¥ Jython unittest ▶ Launch Group ⓓ PyDev Django ಡ PyDev Google App Run	Port number: 10000  Force thread list update on suspend

Figure 24 - Debugger debug configuration menu

Create, manage, and run configu	raiiuns	
		26
🗅 🗈 🗙 🗖 🖶 🔹	Name: vorago_blink_with_BMP	
type filter text	📄 Main 🏇 Debugger 🍉 Startup 🛛 🤤 Source 🔲 Common	
C C/C++ Attach to Application C/C++ Postmortem Debugger C/C++ Remote Application C C/C++ Unit	Reset and Delay (seconds): 3 Hait	
<ul> <li>DSF PDA Application</li> <li>GDB Hardware Debugging</li> <li>vorago_blink_only_ULINK_GDB</li> <li>vorago_blink_with_BMP</li> </ul>	target extended-remote COM5 monitor hard_srst monitor connect_srst disable monitor jtag_scan	~
₽ <sup>I</sup> IronPython Run ₽ <sup>V</sup> IronPython unittest ₽ <sup>V</sup> Jython run	Load Image and Symbols Load image © Use project binary: VA108xc_GCC\Wksp_VOR_Blink\vorago_blink_only\REB1_blink_rom.elf	
¥ Jython unittest ▶ Launch Group ☑ PyDev Django	Ouse file:         Workspace         File System	
by PyDev Google App Run	Image offset (hex):	
💕 Python Run	🛛 Load symbols	

Figure 25 - Startup debug configuration menu

Need to add "connect 1" on line below "monitor jtag\_scan".



# 13 Programming SPI NVM for the VA108x0

There is currently there is no programming support for the GDB environment. We expect to offer this support in the future.

## **14 Conclusions**

This application note has provided all the steps necessary to install and demonstrate GCC using the Eclipse IDE. Two examples were provided for simple and complex projects. You should be able to develop your own projects using the foundation provided.

## **15** Common questions and issues

1. In Eclipse what is the difference between a workspace and a project?

1a: A workspace is the folder / directory containing information in metadata format for the entire project. This includes the path to the project files. A project is a collection of files including: \*.c, \*.h, startup, linker and make files. The project can be stored in the workspace directory or outside of it. Different companies like to organize their projects differently depending upon shared libraries and re-use policies. To reduce confusion, it is recommended that the folder name have useful information with either "workspace" or "project" being part of the name.

2. When starting a new project, should I import an older one or use "new - > makefile from existing"?

2a. Starting new project from scratch can be a long and laborious task. Copying the entire workspace of the REB1 project and renaming it is a good way to start. Otherwise using the "new -> makefile from existing" option is the second option.

3. Inside the debug window, when and why use "relaunch"?

3a. If debugging code included modifying memory or registers, it normally a good idea to reset the part and to reload the program periodically. The relaunch command will reset the MCU and download the active project to Instruction RAM on the VA108x0.



4. Can MCU peripheral registers be viewed by the debugger?

4a - This feature has not been implemented at this time. VORAGO plans on adding this feature in the future.

5. Is it possible to program SPI memory from the debugger?

5a - Ans: Yes this is possible but support for it is not available at this time. VORAGO plans on adding this feature in the near future.

6. The MCU will not respond to any interrupt requests like it had in previous debug sessions. What is going on?

6a. The MCU can react to a hard fault problem in a way that is not recoverable until the unit is powered down. To check for this condition, view location 0xE000ED04. If the least significant 11 bits are 0x003, the device is in this condition. Most likely the cause of this is code trying to access invalid memory space. To correct the condition a POR reset is required.

7. I have modified the source code in the edit window, but the compiled output does not reflect the changes. Why is that?

7a. Eclipse will always use the saved files and not the version you are viewing. Save the file and rebuild the project to remedy this issue.

# **16 Other Resources**

VORAGO VA108x0 programmers guide & VORAGO MCU products: http://www.voragotech.com/VORAGO-products

VORAGO Application notes: <u>http://www.voragotech.com/resources</u>

VORAGO VA108x0 REB1board user guide: Part of Board Support Package (BSP) <a href="http://www.voragotech.com/products/reb1">http://www.voragotech.com/products/reb1</a>

MCU on Eclipse – Excellent reference for Eclipse <u>https://mcuoneclipse.com/category/eclipse-</u> 2/

#### **Revision log:**

June 11, 2018 – Initial release

April 2, 2019 - Rev 1.1

- Eclipse support for Reset command to debugger not working. Need to add work around.
- Better explanation of workspace
- Minor corrections and improvements to diagrams.
- Screen captures of latest download pages.