REB1 USER MANUAL

Evaluation board for VA108x0 MCU from VORAGO

FEBRUARY 10, 2017

VORAGO TECHNOLOGIES

Contents

1 Introduction

1.1 Purpose of Document

This document is intended to provide instructions on how to use all features of the REB1 evaluation board and provide a working platform for software development with either the VA10800 or VA10820 MCUs from VORAGO. The VA10820 has supplemental EDAC (error detection and correction) and memory scrub functionality. In all other ways, the two devices are register set identical.

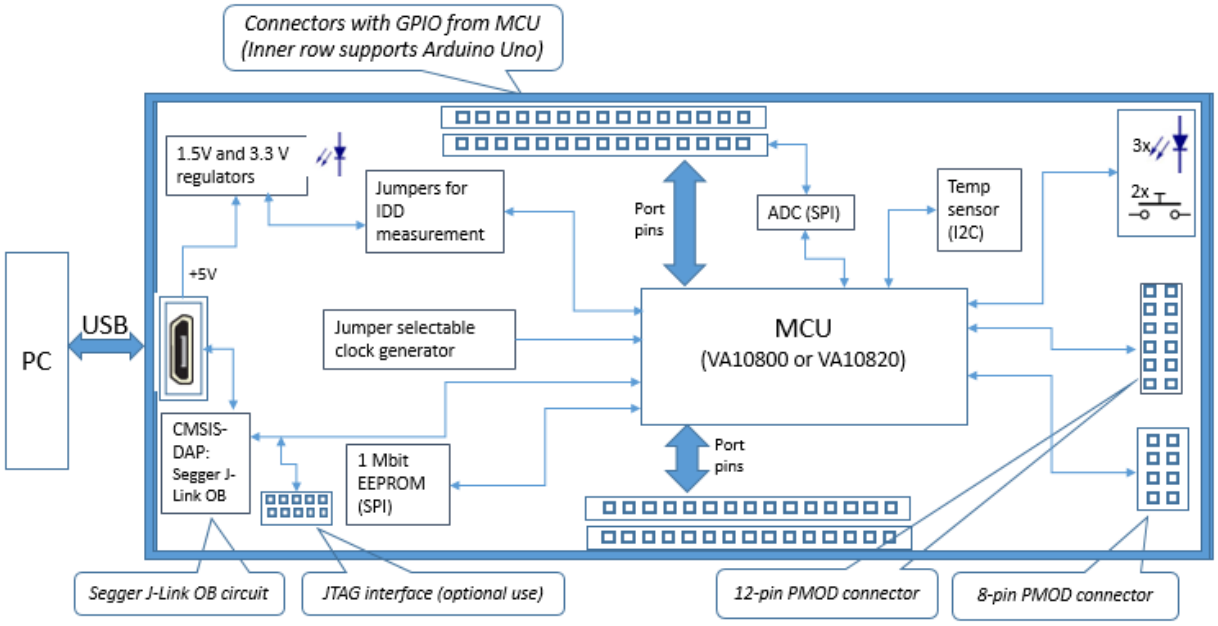
1.2 Overview of Hardware and Software components

The REB1 evaluation board includes a JTAG interface from Segger known as J-Link OB. There is no need for a separate JTAG debug pod although a connector is provided if a different tool is preferred. SEGGER Microcontroller is a full-range supplier of hardware and software development tools. The J-Link OB allows the board to be connected directly to the USB port of a PC to allow:

- Power to be supplied from the USB 5V supply
- JTAG communications for debug and programming
- Terminal communications to allow data transfer between a PC terminal window and the VA108x0 MCU.

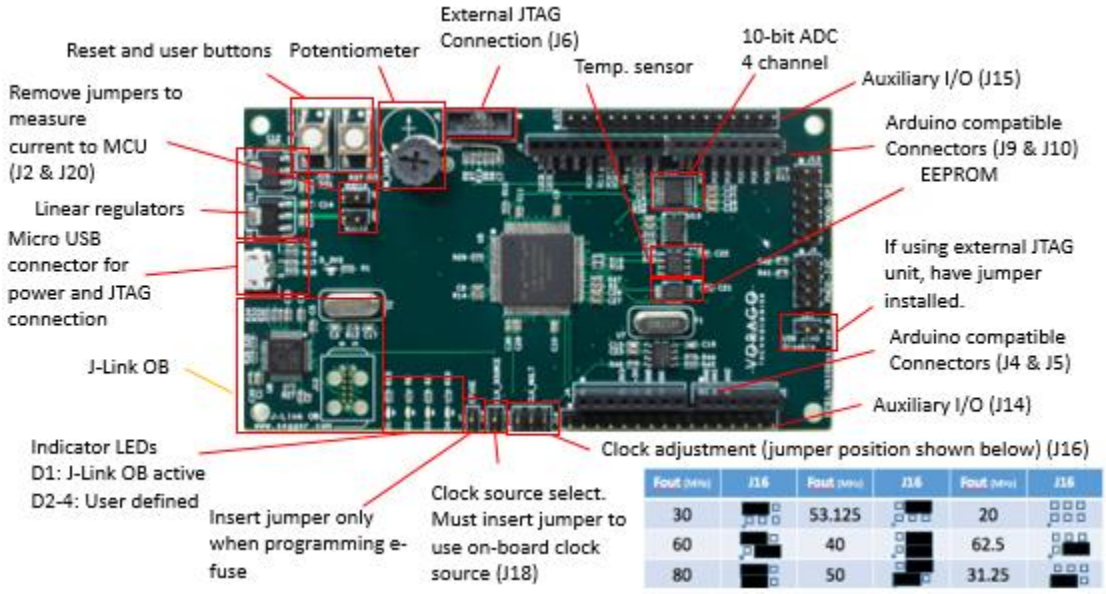
A block diagram of the evaluation board is shown here. The intent of the board is to provide sufficient hardware to evaluate all facets of the MCU at room temperature.

Figure 1 - Block diagram of REB1



1.3 REB1 board component placement diagram

Figure 2 - Photo of REB1 with functional blocks identified



1.4 Connector pin assignment table

The schematic for the board is one of the included files in the REB1 software download package. To assist with quickly finding which pins are tied to the various connectors on the board, the following set of tables are provided.

Table 1 - REB1 Connector designations

J4	Power connect
1	NC
2	NC
3	NC
4	VCC_SYS
5	+5V
6	VSS
7	VSS
8	NC

J5	ADC connect
1	AN0
2	AN1
3	AN2
4	NC
5	NC
6	NC

J8	MCU connect
1	I2CB_SCL
2	I2CB_SCL
3	I2CB_SDA
4	I2CB_SDA
5	VSS
6	VSS
7	VCC_SYS
8	VCC_SYS

J8 layout

J9	MCU connect
1	I2CB_SCL
2	I2CB_SDA
3	VCC_SYS
4	VSS
5	PORTA_31
6	PORTA_29
7	PORTA_30
8	PORTA_28
9	PORTA_23
10	PORTB_22

J10	MCU connect
1	PORTA_0
2	PORTA_1
3	PORTA_2
4	PORTA_3
5	PORTA_4
6	PORTA_5
7	PORTA_9
8	PORTA_8

J13	MCU connect
1	PORTA_23
2	PORTA_30
3	PORTA_29
4	PORTA_31
5	VSS
6	VCC_SYS
7	NC
8	NC
9	NC
10	NC
11	VSS
12	VCC_SYS

J13 layout

J14	MCU connect
1	PORTB_10
2	PORTB_11
3	PORTB_12
4	PORTB_13
5	PORTB_14
6	PORTB_15
7	PORTB_16
8	PORTB_17
9	PORTB_18
10	PORTB_19
11	PORTB_20
12	PORTB_21
13	VCC_SYS
14	VCC_SYS
15	VSS
16	VSS

J15	MCU connect
1	PORTA_27
2	PORTA_26
3	PORTA_25
4	PORTA_24
5	PORTB_2
6	PORTB_3
7	PORTB_4
8	PORTB_5
9	PORTB_6
10	PORTB_7
11	PORTB_8
12	PORTB_9
13	VCC_SYS
14	VCC_SYS
15	VSS
16	VSS

1.5 Materials List

- REB1 evaluation board – Programmed with demonstration program
- 36" Micro USB cable
- Insert card with component placement picture and URL
- 7 jumpers.

1.6 Support

Email: info@voragotech.com Reference REB1 in the email title.

2 Software Setup

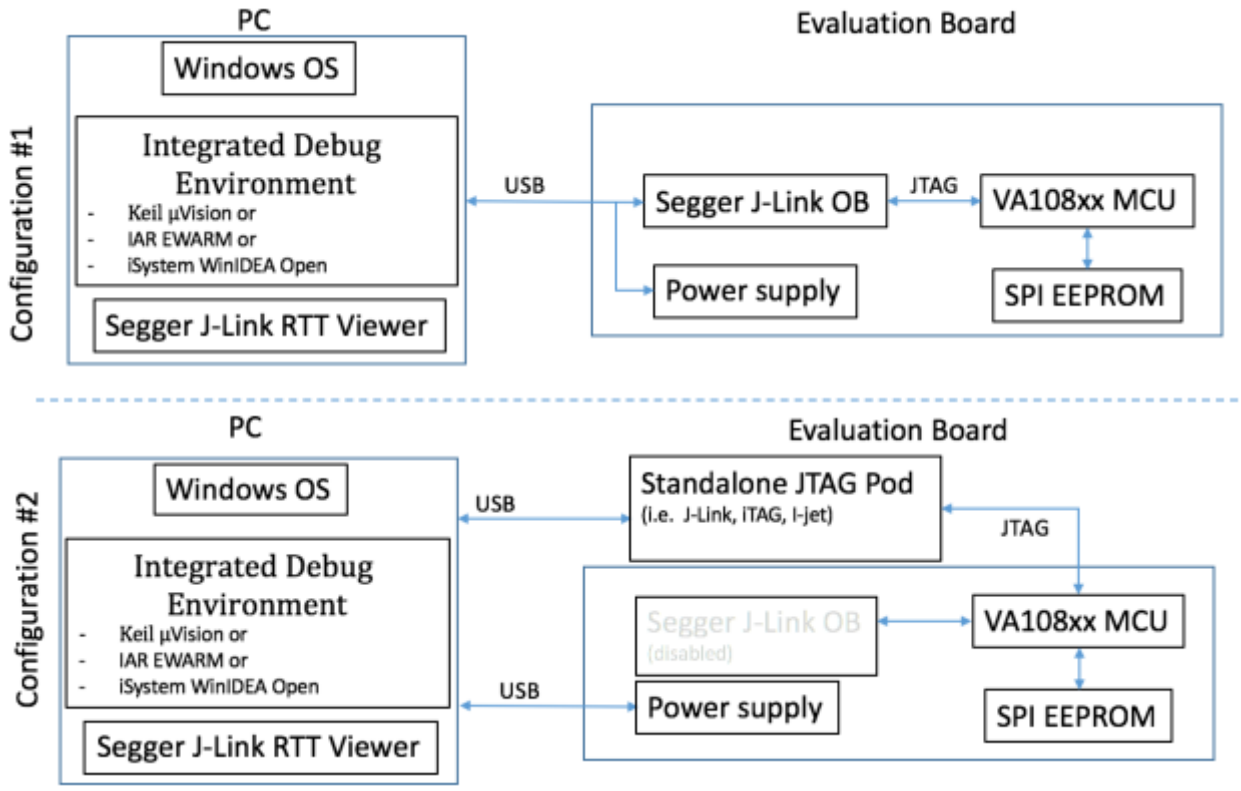
Three sets of software will be working in concert:

- Integrated Development Environment (IDE) {Either Keil, IAR or iSystem},
- Segger J-Link RTT viewer and
- Embedded code running on the VA108x0 processor.

The IDE allows code to be developed, compiled, debugged and programmed to an SPI EEPROM on the board. The J-Link RTT Viewer allows terminal communication between the PC and the VA108x0 MCU. Keys pressed on the keyboard can be routed to the VA108x0 MCU and printf

statements from the VA108x0 MCU can either be logged into a file or shown on a terminal window. See Figure 3 - Software interaction for REB1.

Figure 3 - Software interaction for REB1

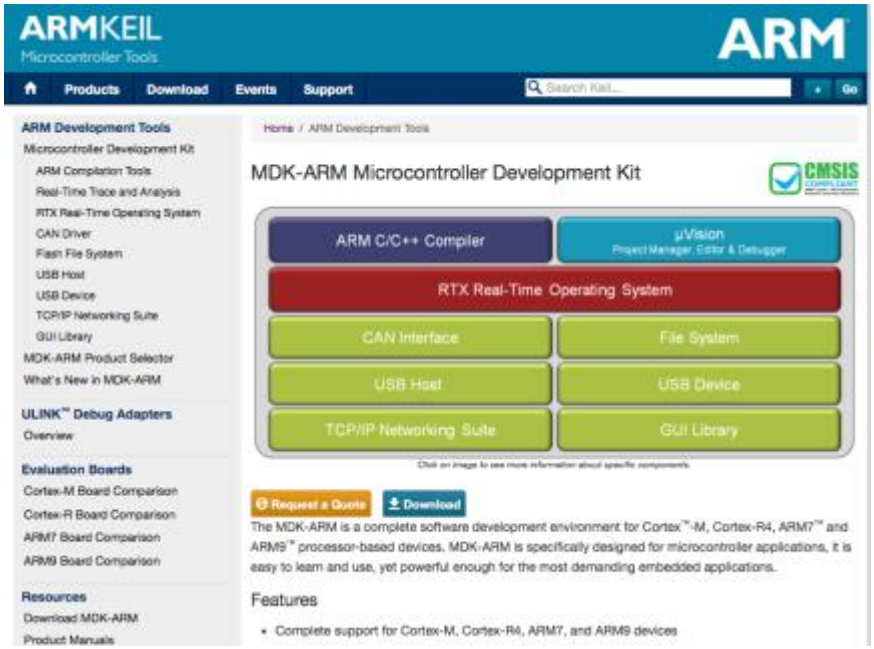


2.1 Required Downloads

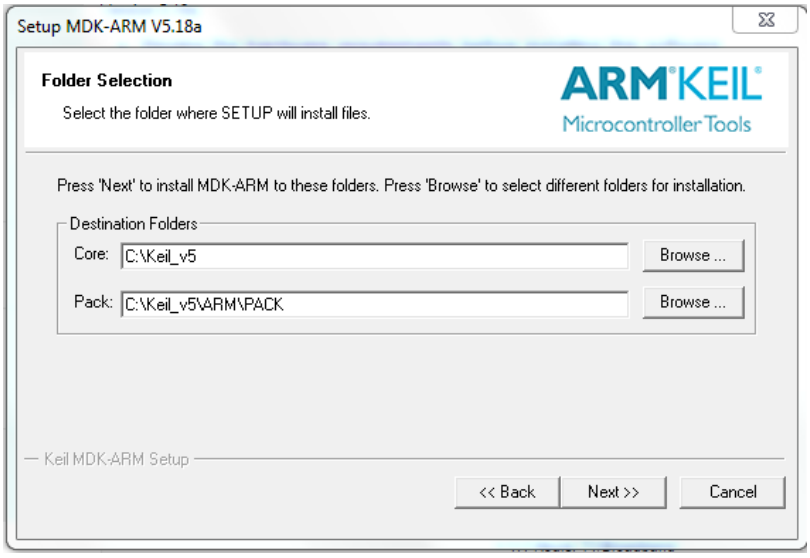
The following downloads are required to allow full functionality of the evaluation board. These tools are evaluation versions and are free of charge. Three IDE options are provided. Only one IDE is needed. Please download one IDE, the Segger J-Link software and the VORAGO REB1 software development code as outlined in the below sections.

2.1.1 Keil μVision Integrated Development Environment

Keil offers an evaluation version of the IDE for free that will support compiled code of 32kbytes and below. http://www.keil.com/arm/mdk.asp (approximately size = 400 Mbytes)



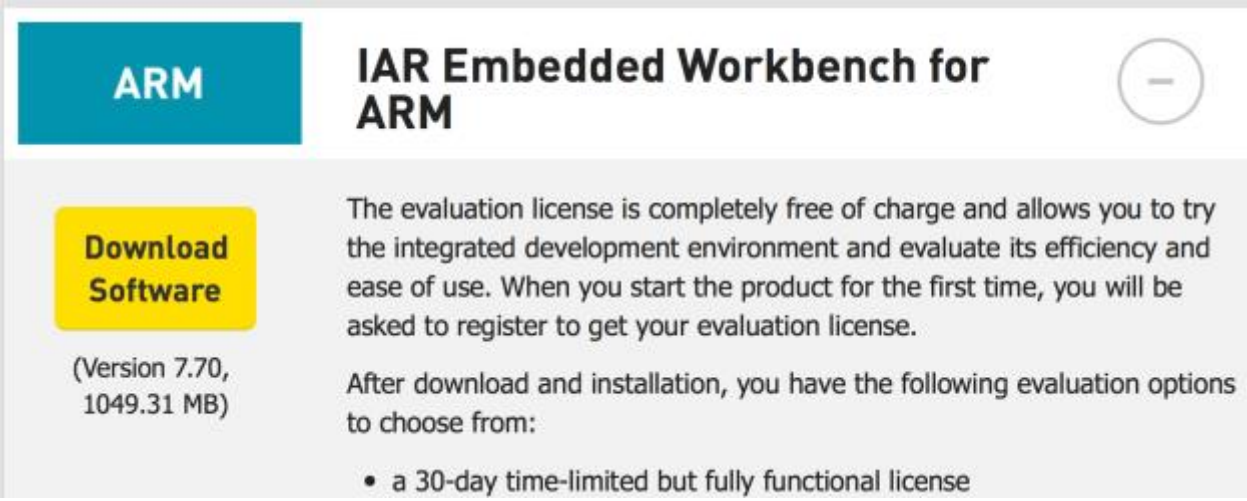
Please use the default directories for the "core" and "pack" when prompted for information during the install.



Before the MDK install is complete, the pack installer dialogue will open. Let it run to completion and then close it. The va108xx.pack will be loaded later when we open the project.

2.1.2 IAR Embedded Systems for ARM IDE

Visit https://www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials and download the latest version. At the time of this document creation, the latest version was 7.7. The download size is approximately 1GB.



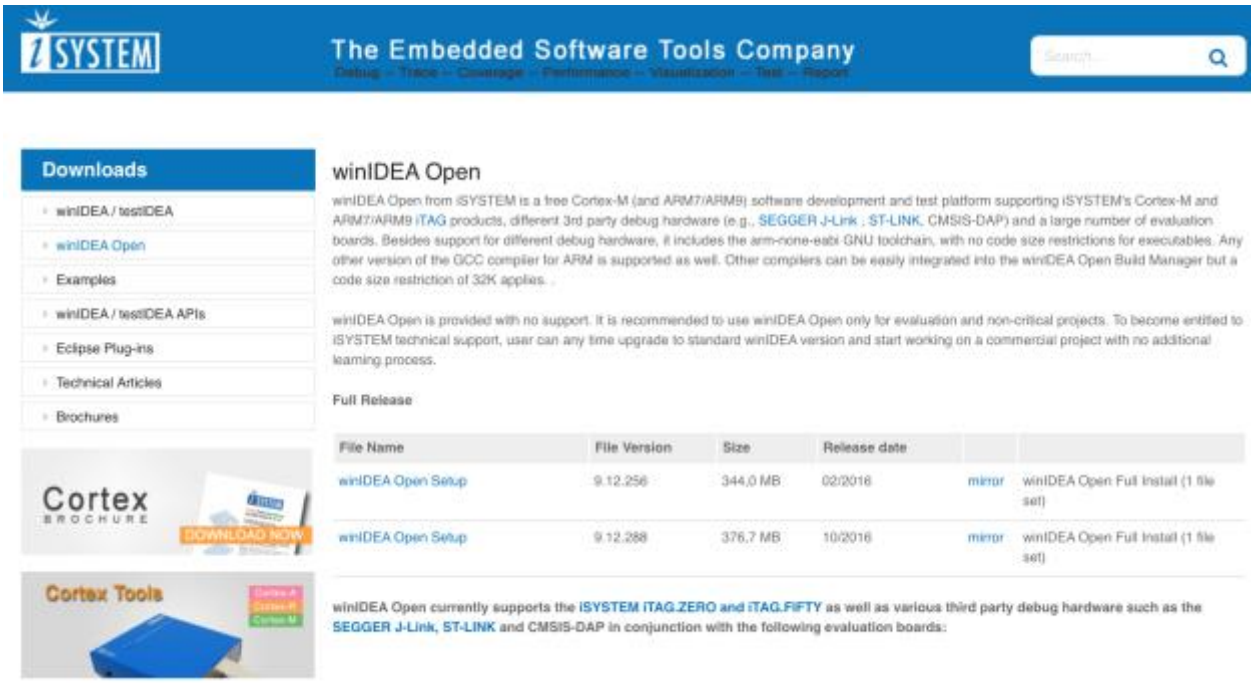
Please use the default install directories. The IDE will be invoked in a later step when the example project is started. Please select the 30-day time-limited version of the code. Most projects for the VA108x0 parts exceeds the 16 kbyte limit of the perpetual free version.

2.1.3 iSystem winIDEA

Visit http://isystem.com/products/software/winidea and navigate to the download page http://isystem.com/download/winideaopen. The latest at the time of printing this document can be found at http://www.isystem.com/downloads/winIDEA/setup/winIDEAOpen9_12_294.exe Down load the software and install it as instructed. Use default paths. Important files will be stored at: …/user/"YourUserName"/AppData/Roaming/ASYST/WinIDEA. There should be 5 files unique to the VA108xx in this folder:

- VA10820.sfr
- VA10800.sfr
- VA108xx.svd (contains all register definitions for the device)
- UMI2_va108xx_EE.s32 (programming file that the iTAG50 tools uses to program the SPI EE on the REB1 board.
- VA108xx.json (contains debug interface specifics for a device)

If these files are not present, please copy and paste them from the iSystem project folder. /REB1_BSP.software/mcu/projects/reb1_va108xx_iSystem/app/reb1_reference/reb1_va108xx.



2.1.4 Segger J-Link

https://www.segger.com/jlink-software.html
Select Software and documentation pack for Windows V6.10n [23,162 KB]. If a later version is available, please use it.

2.1.5 VORAGO Software and Documentation

http://voragotech.com/REB1.
This will have the following components in a single .zip file:

- VA108x0 Software Development kit (Folder structure with many *.c, *.h and Project files)
- VA108x0 Software Documentation (html file structure)
- VA108x0 Pack file with programming algorithms, header and .SVD files. (Single .zip file that can either be open separately or by the Keil IDE)
- REB1 Schematic (Single .pdf document)
- REB1 User's Manual (Single .pdf document)
- REB1 Quick Start Guide (Single .pdf document)

3 Hardware check

3.1 Powering up the board

The Segger J-Link driver needs to be loaded prior to plugging the board to a PC. See 2.1.4.

- Before connecting the board conduct a jumper check. Only the following jumpers should be inserted:
  - Clock multiplier select (J16). Two jumpers as shown in figure 2.
  - Clock source jumper (J18)
  - MCU voltage supply shunts (J2 & J20)
- Connect the USB cable between PC and the REB1 board
  - The D_3V3 LED will indicate that power is applied.
  - D1 will indicate that the J-Link OB enumerated and has successfully connected to the VA108x0 device. If D1 does not turn on, or continually flashes, remove jumper J21.
- If the MCU has the pre-programmed example code running, LED D2 will blink at a relatively fast rate of 10 Hz.
  - Pressing the **RESET** button (S2) will hold the device in RESET. Releasing it will commence the boot sequence and start code executing.
  - Pressing the user switch (SW_USER) will toggle LED D3 on and off. If the button is held low, the LED will toggle on and off.

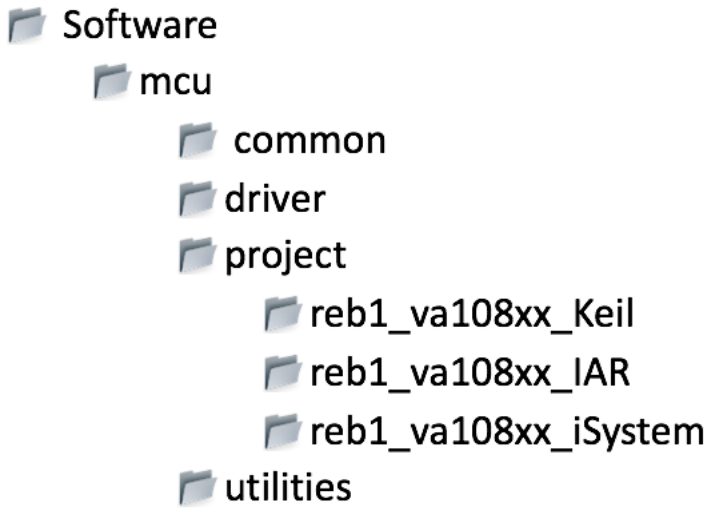
4 Starting an IDE and building a program

Depending upon which IDE you decide to use, the steps to run a program will be different. The following three sections provide step by step instructions for starting the IDE, downloading code and running a program on the VA108x0.

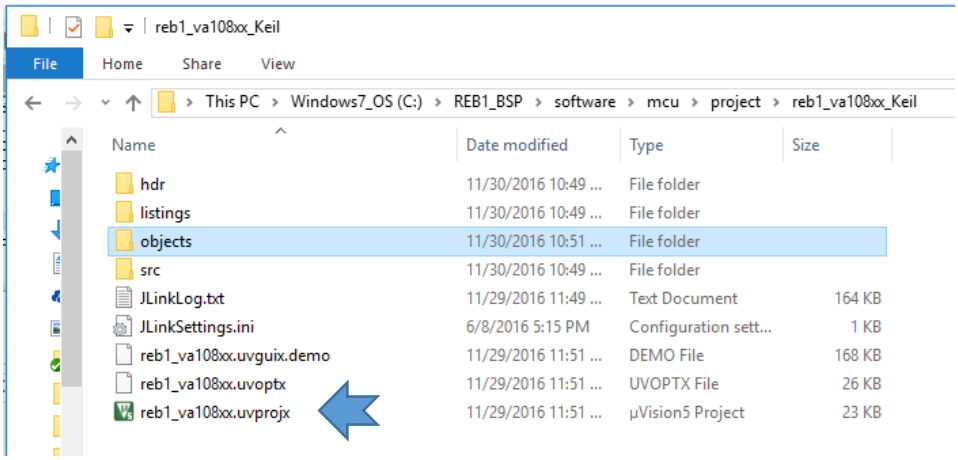
4.1 Keil IDE –

4.1.1 Opening a project

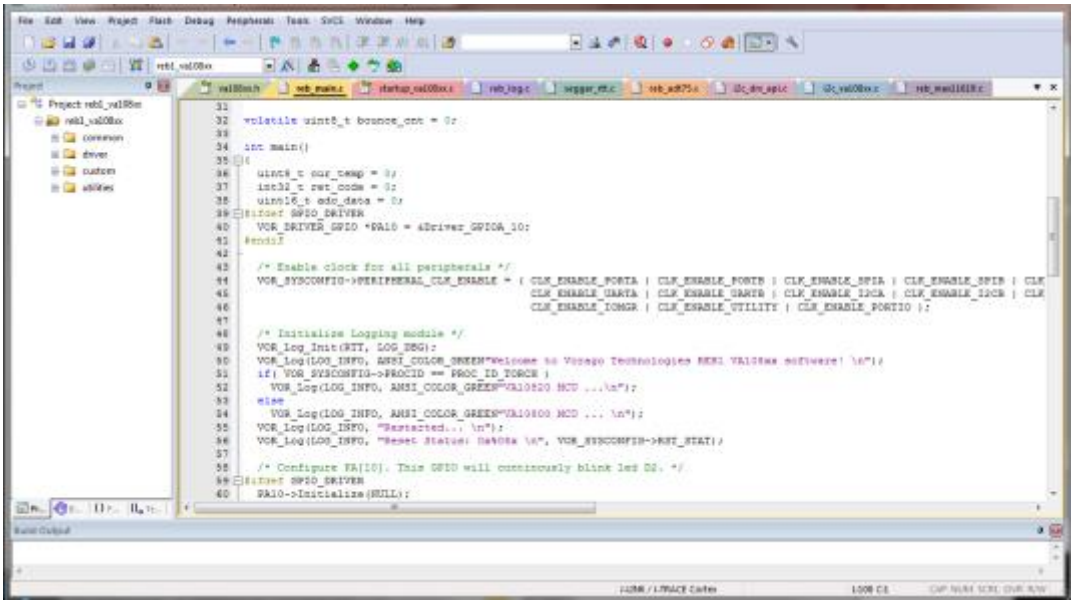
The download from VORAGO has a file structure as outlined here.



A populated Keil project is available in the ../software/mcu/project/reb1_va108xx_Keil folder as shown here.



Double click on the file ending with uvprojx and the Keil IDE should open with this project loaded. The screen should look like the figure below.



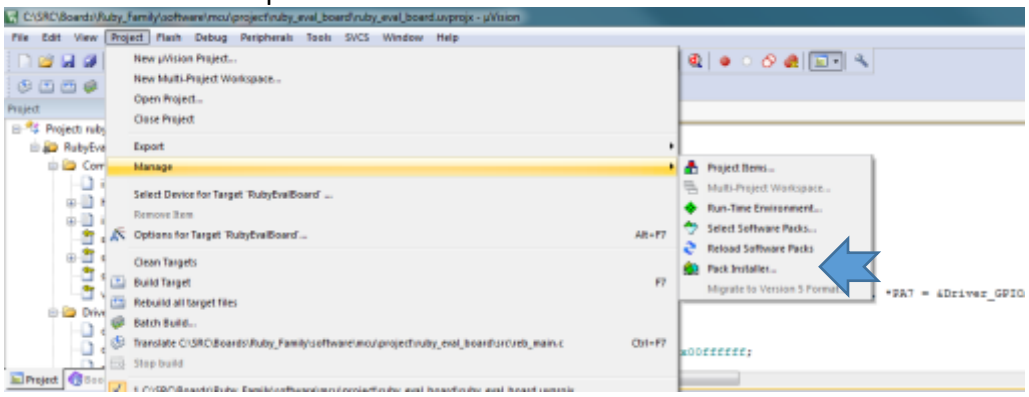
It is possible that you may get an error message referencing an invalid device being selected. This can be expected since the device pack file for the va108x0 was not installed yet. Continue to the next section to install the pack file.

4.1.2 Installing a Pack File

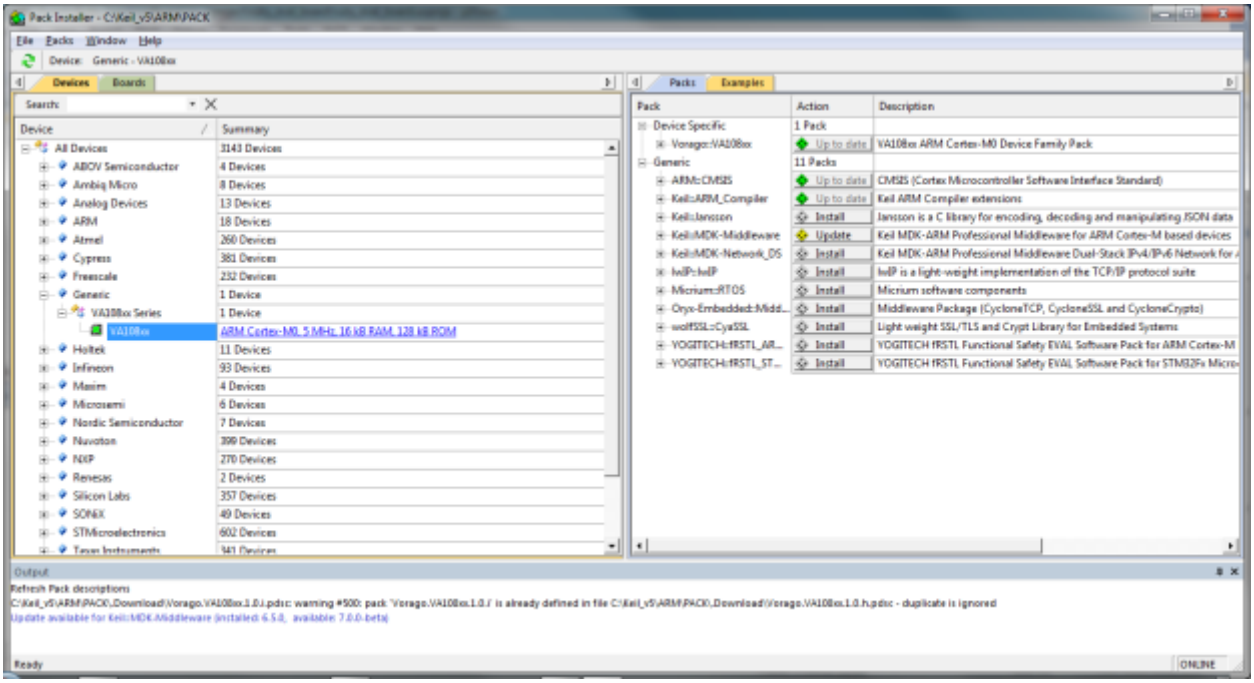
As part of the multi-level CMSIS standard (http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php.), ARM has created an efficient way to pull in all the necessary information for an IDE to work with an MCU. The "pack" file has: NVM programming code, header file information, documentation for the MCU, and an SVD file (System View Description).

To import the pack file, follow these steps:

- From the Project pull-down menu, select "manage" and then "pack installer". This will open another window.



- From the pack installer window, Use the "file" pull-down menu and select "import". Navigate to the VA108xx1.0.0.pack file which was part of the VORAGO download and hit the **Open** button.



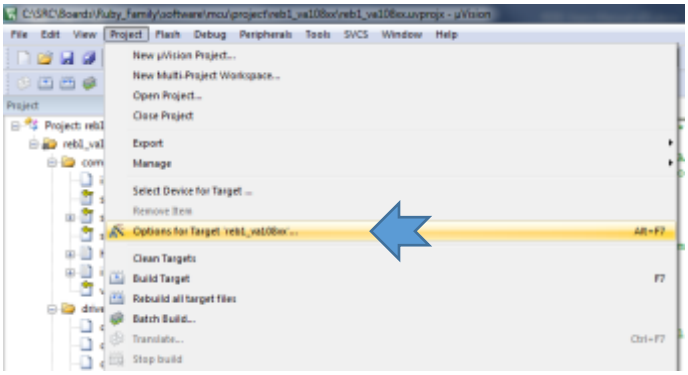
At this point, the VA108x0 information will be installed in the Keil file structure and will be available for use. It includes:

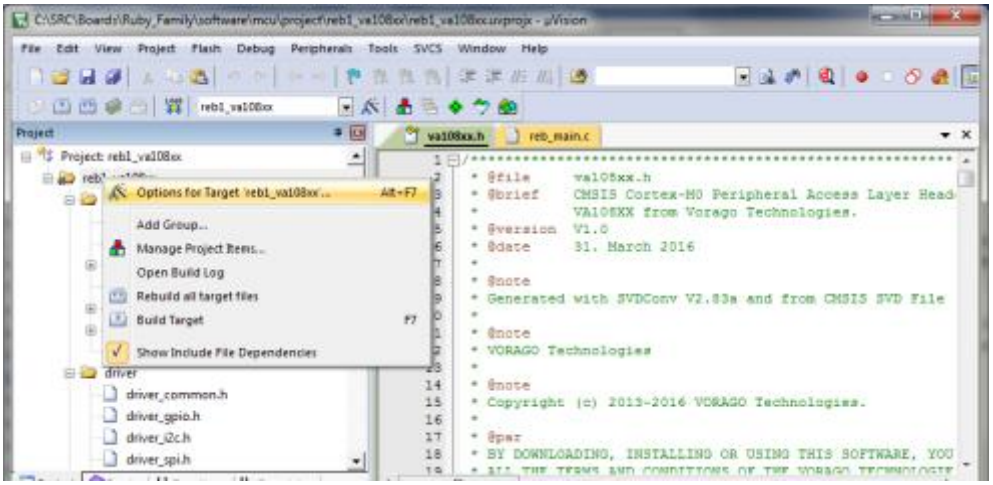
- Datasheet and programmers guide
- SVD file (used by debugger to show register and bit names)
- Va108xx.h file which has all the register and bit definitions
- Programming files for the SPI EE on the board and other SPI memories.

4.1.3 Configuring options

The Keil IDE is a very powerful and flexible tool which requires several options to be assigned before it is operational with a specific MCU and JTAG debugger probe. The following sections provide the minimum options for the VORAGO software development kit to function.

All the options can be accessed under the **Project** pull-down menu as shown here. An alternative way to access the options is to right-click on top level folder of the project.

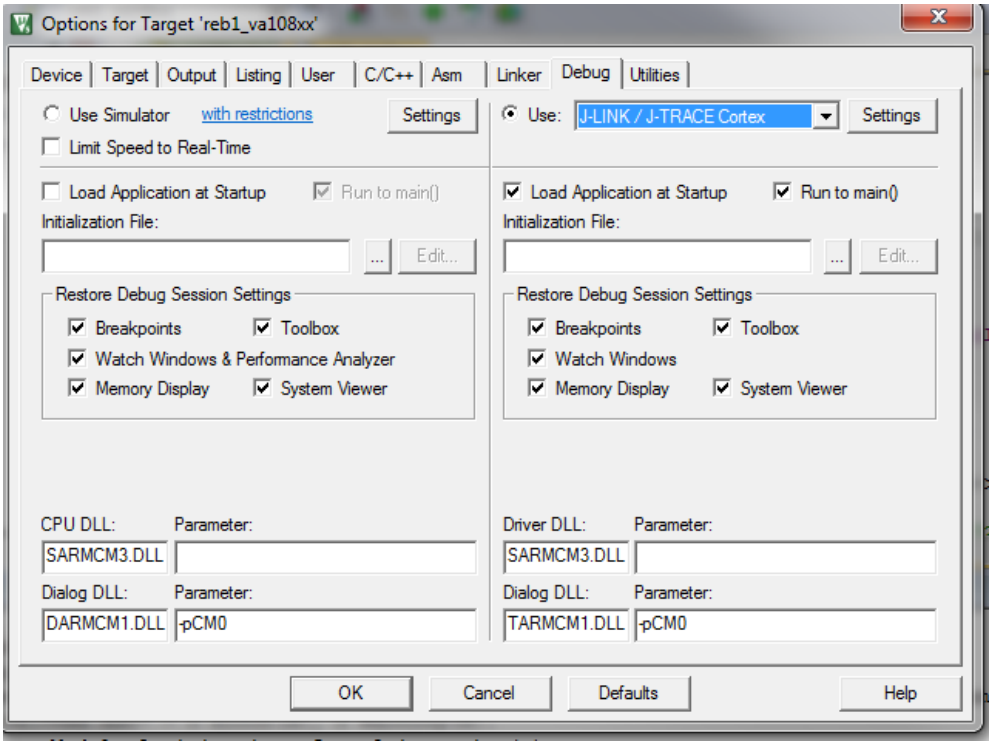




An options window with 10 pull-down menus will open.

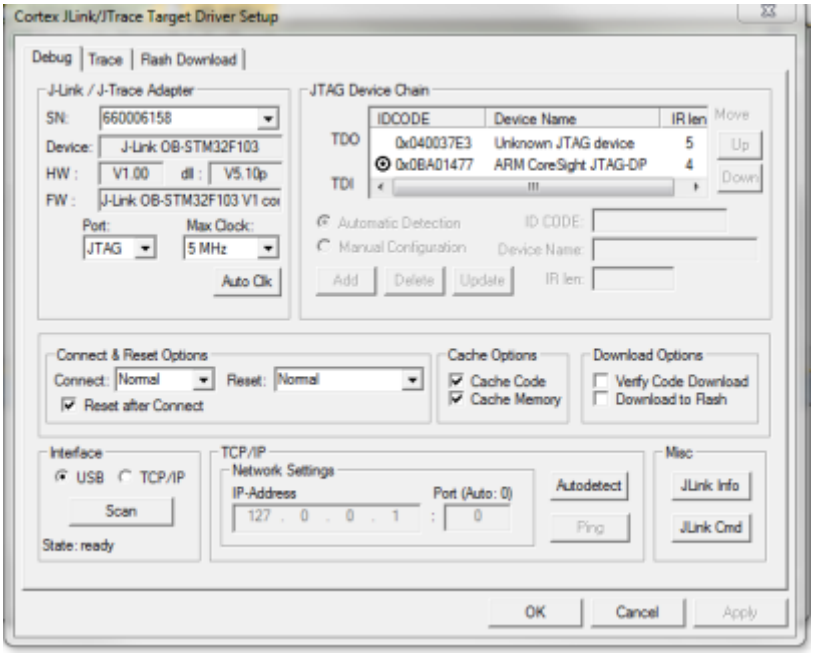
4.1.4 Debug options

The Keil IDE supports many forms of JTAG interfaces. The default debug connection is the ULINK2 from Keil. The REB1 board has a built in JTAG interface called Segger J-Link OB. The IDE must be told which JTAG interface is used. See below for screen captures on how to do this. First use the pull-down menu of the **Debug** options window to select "J-LINK/J-TRACE Cortex". Note that there is no special selection for J-LINK OB.



Second, set the J-Link to use the JTAG connection. Click the **Settings** button and select "JTAG" in the Port Window. The "Max Clock" selection can be anything up to 5 MHz. The board should be connected to a PC prior to setting up the Debugger. This will allow the tool to identify the J-LINK OB and the IDCODEs of the MCU's TAP controllers.

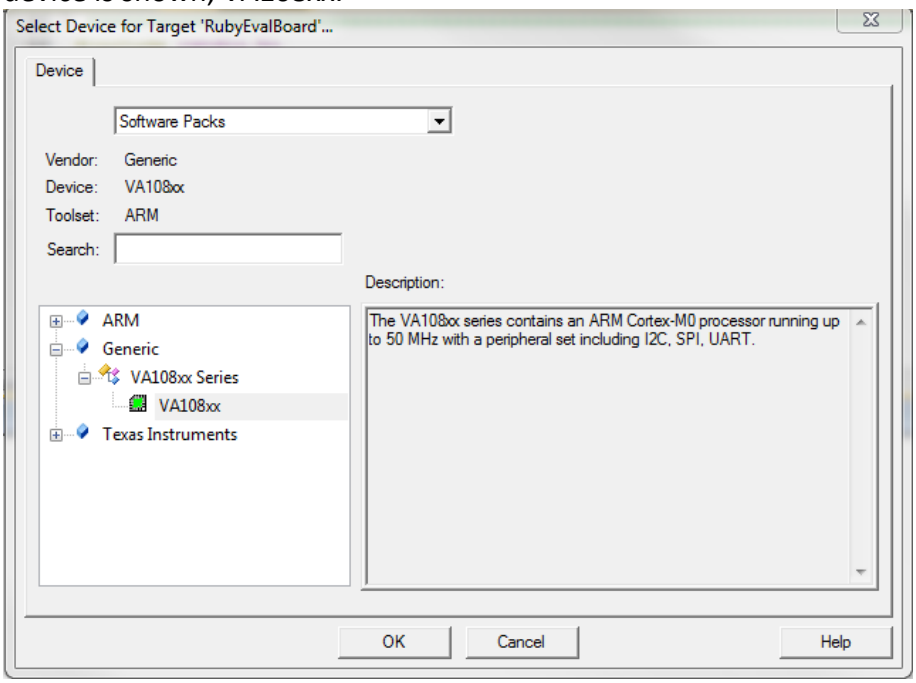
Note: An error message may be displayed stating the J-Link does not recognize the MCU. If that occurs, hit "ok" and select the generic Cortex M0 in the subsequent dialogue box. This error message should only occur on the first attachment to the board with the MDK.



Note that two IDCODEs will appear in the "JTAG Device Chain" window. One is the standard ARM CoreSight JTAG-DP, the other is the test chain for the device.

4.1.5 Select device

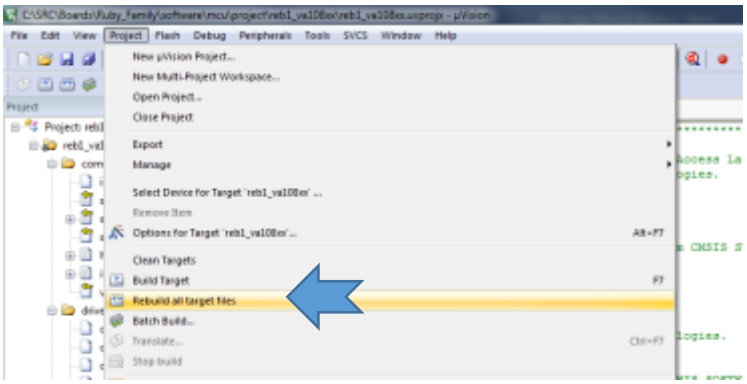
If the Vorago.VA108xx.vers.pack file has been imported, the VA108x0 will be shown under the generic group. The memory map for the VA10800 is a subset of the VA10820 and only one device is shown, VA108xx.



4.1.6 Project Build

The Keil tool has several ways to access many functions. There is always a pull-down menu available but many functions have hot keys or icons that can be clicked on. To compile and link the entire project, the "Rebuild all target files" function must be called. The **Build** button translates modified or new source files and generates the executable file. The **Rebuild** command translates all source files regardless of modifications. Options for doing this include:

- Pull down menu

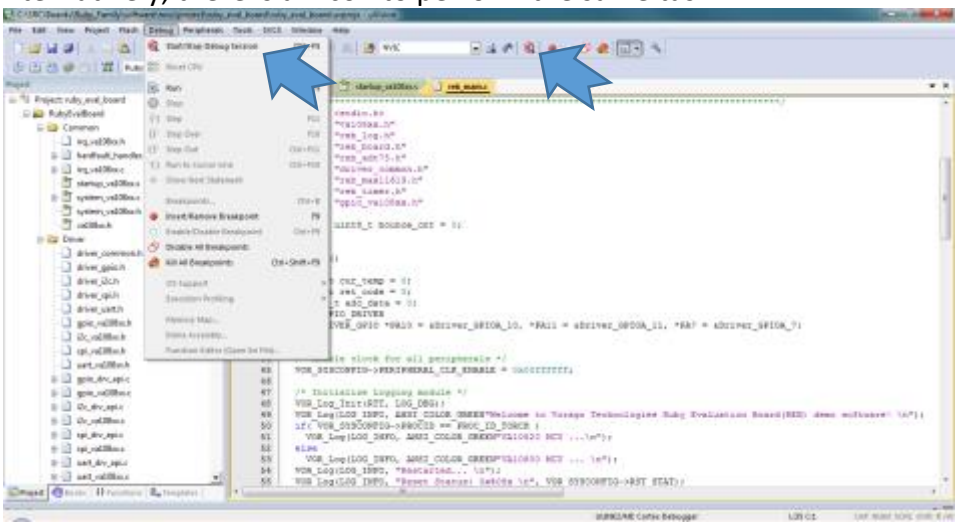


- Icon

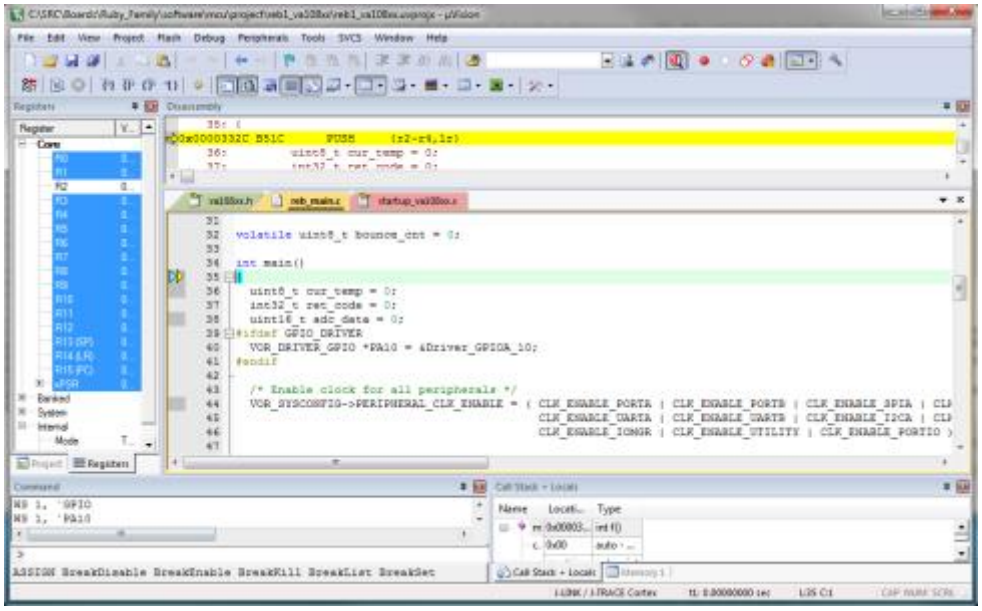
Note: The "F7" is a hot key for the **Build** Button**.**

4.1.7 Download and debug

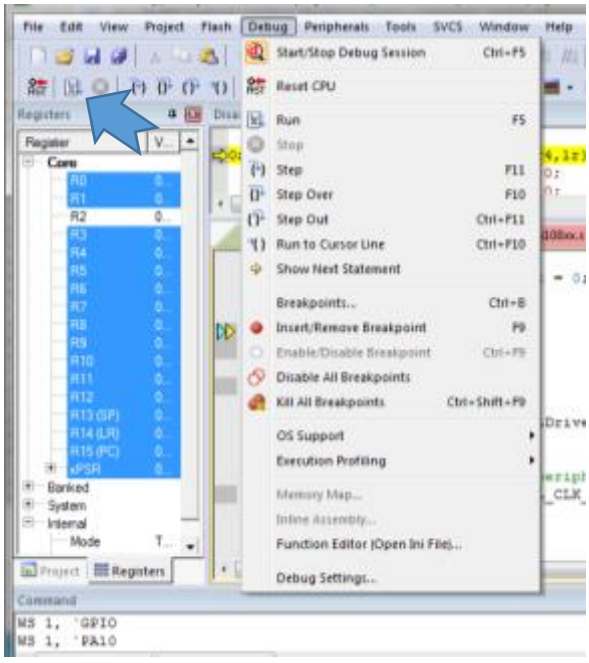
To enter a debug session, use the **Debug** pull-down menu and choose "Start/Stop Debug Session". Alternatively, there is an icon to perform the same task.



Once the IDE has entered debug mode, the screen appearance will resemble the following image.



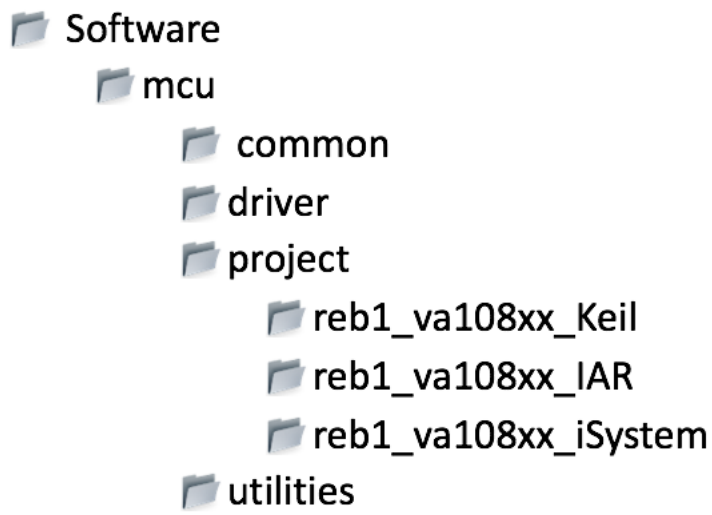
Many options are available under the debug menu as shown here. Most of these functions have buttons on the menu bar for quick point and click access. If the demonstration program has been loaded, the **Run** button can be pressed to begin code execution. LED D2 should begin to blink.



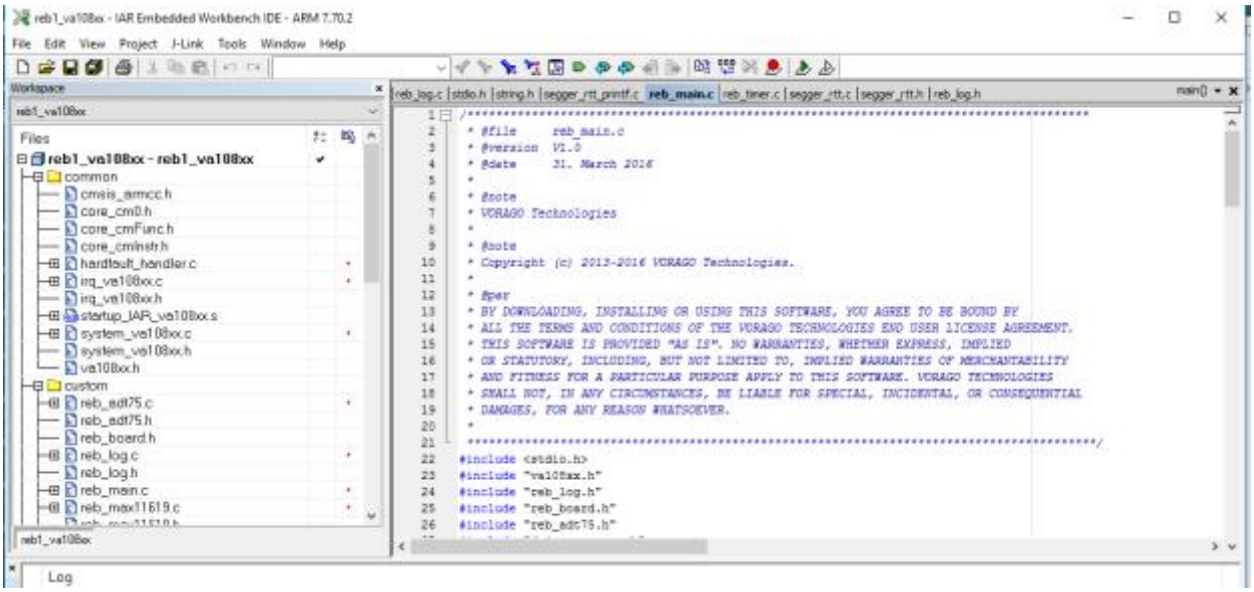
4.2 IAR EWARM IDE –

4.2.1 Opening a project

The BSP download from VORAGO has a file structure as outlined here. There are several unique files for IAR that are stored inside the reb1_va108xx_IAR folder. Common, driver and utilities folders contain files that work with any IDE.



Inside the reb1_va108xx_IAR folder is a workspace file, reb1_va108xx.eww. Double click on this file and IAR EWARM tool will open with the REB1 software loaded.

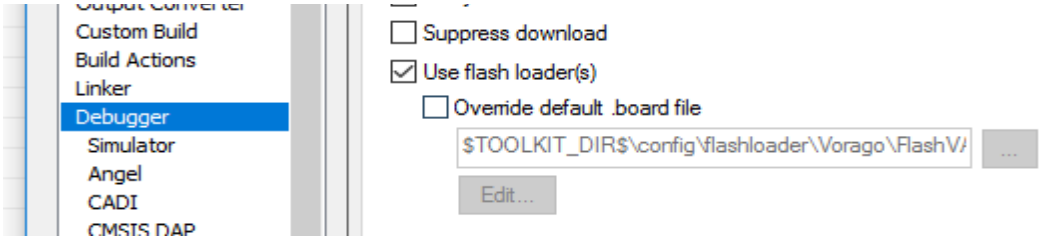


4.2.2 Select device

To set the device, select the **Project** pull-down and follow the selection path as shown here:

Project -> Options -> General Options –> Device -> "Vorago VA10800"

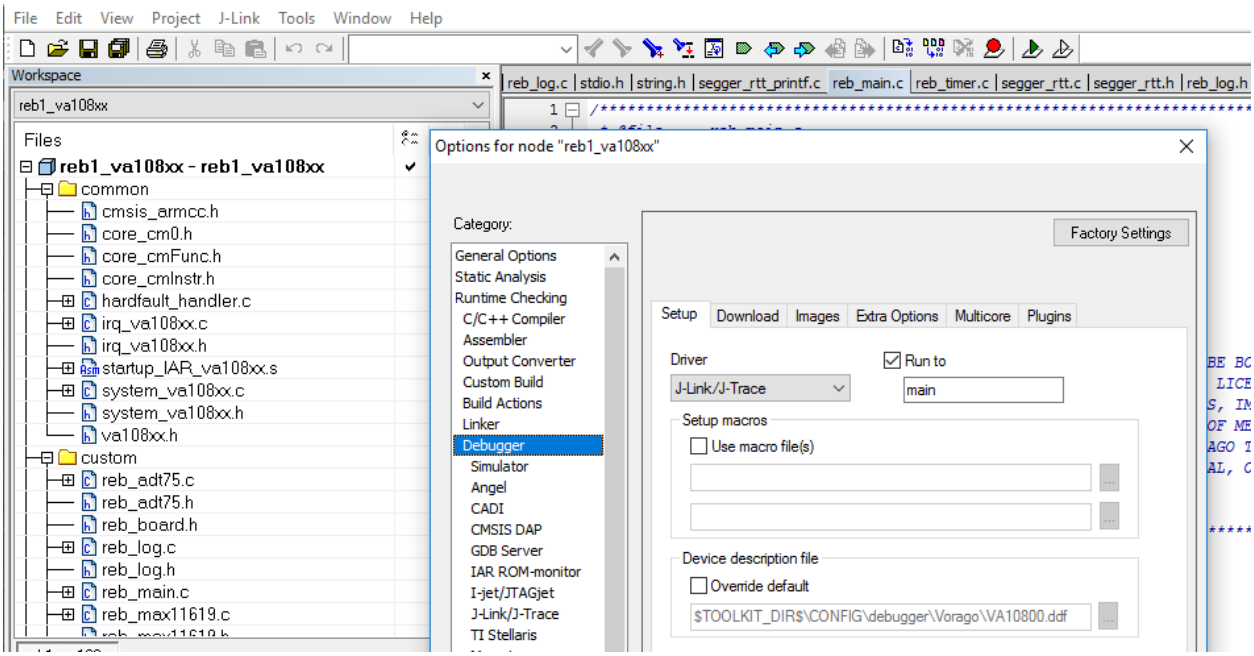
The linker and flash loader should automatically fill when the device is select.



4.2.3 Configuring options

The tool has many configuration options. Only the critical ones are discussed here. The IDE must know which debug pod is being used. Since the REB1 board comes with an on-board Segger debugger, we need to select "J-Link". Under the **Project** pull-down menu, select **Options** and then **Debugger.** Set the driver to "J-Link / J-Trace".

Project -> Options -> Debugger -> Setup -> J-Link/J-Trace



4.2.4 Clean and Project Build

Before a program can be executed, it must be built which includes compilation and linking. Prior to rebuilding the project, it a good idea to "clean" the workspace which can remove stale files. Using the **Project** pull-down menu, select **Clean**.

Project -> Clean

To rebuild, use the **Project** pull-down and select **Rebuild All**.

Project -> Rebuild All

4.2.5 Download and debug

IAR offers many options for entering debug mode. We are only going to explain how to use one of those. Under the **Project** pull-down, select **Download and Debug**.

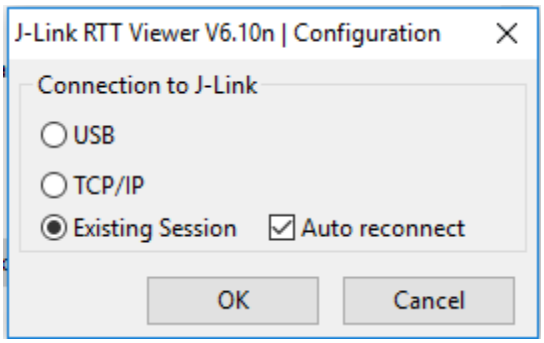
Project -> Download and debug

4.2.6 Running the program

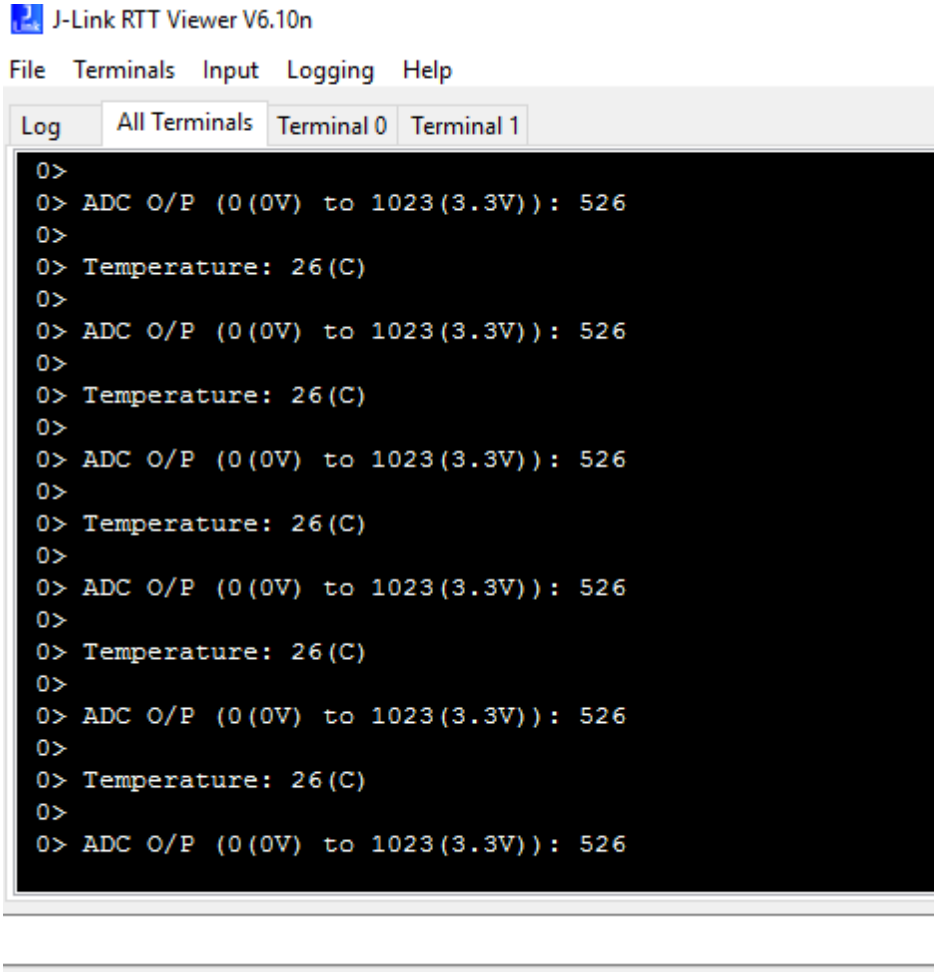
To begin program execution, use the **Debug** pull-down and select **Go**. If the program is running properly, LED D2 will be flashing.

4.2.7 Using Segger RTT (Real Time Terminal)

Section 4.4 provides more detailed information on the RTT. Specific to the code created for the IAR tool, the connection procedure is a bit different. First follow the steps in 4.2.1 to 4.2.6. At that point, the software is running on the MCU and it is possible to connect with the RTT. Navigate to the RTT_Viewer application (Program Files (x86)s -> Segger -> JLink_V6.10n -> JLinkRTT_Viewer.exe). A dialog box like the one shown here will appear. Select "Existing Session" and "Auto reconnect", then click on OK.



The temperature and ADC information should be broadcast to the terminal every 5 seconds as shown here.

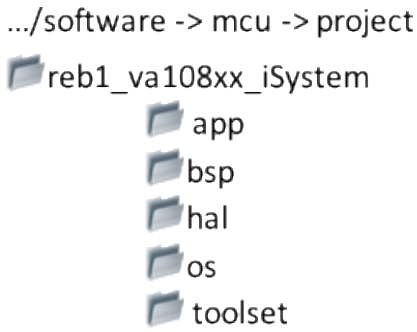


4.3 iSYSTEM winIDEA IDE –

winIDEA is an open platform that supports several JTAG probes such as Segger and the iSystem iTAG50. Programming support of SPI memory devices with the VA108xx is only available with the iTAG50 probe which is available from Amazon(amazon.de).

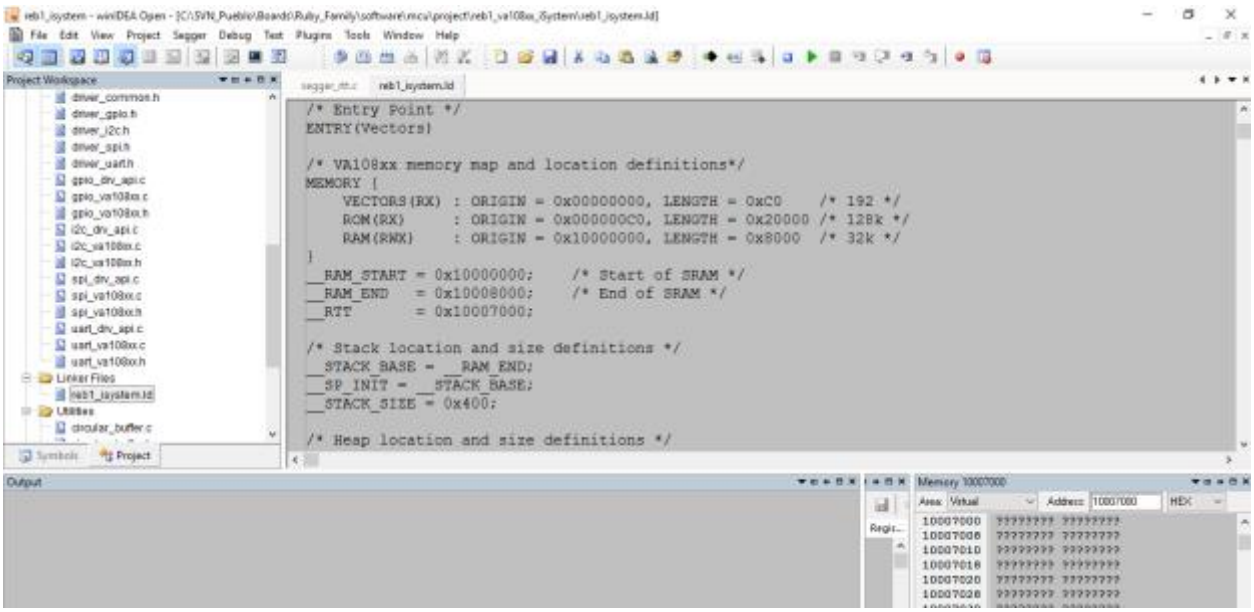
4.3.1 Opening a project

The download from VORAGO has a file structure as outlined here. A similar project is available from the iSystem website.: http://www.isystem.com/download/examples.



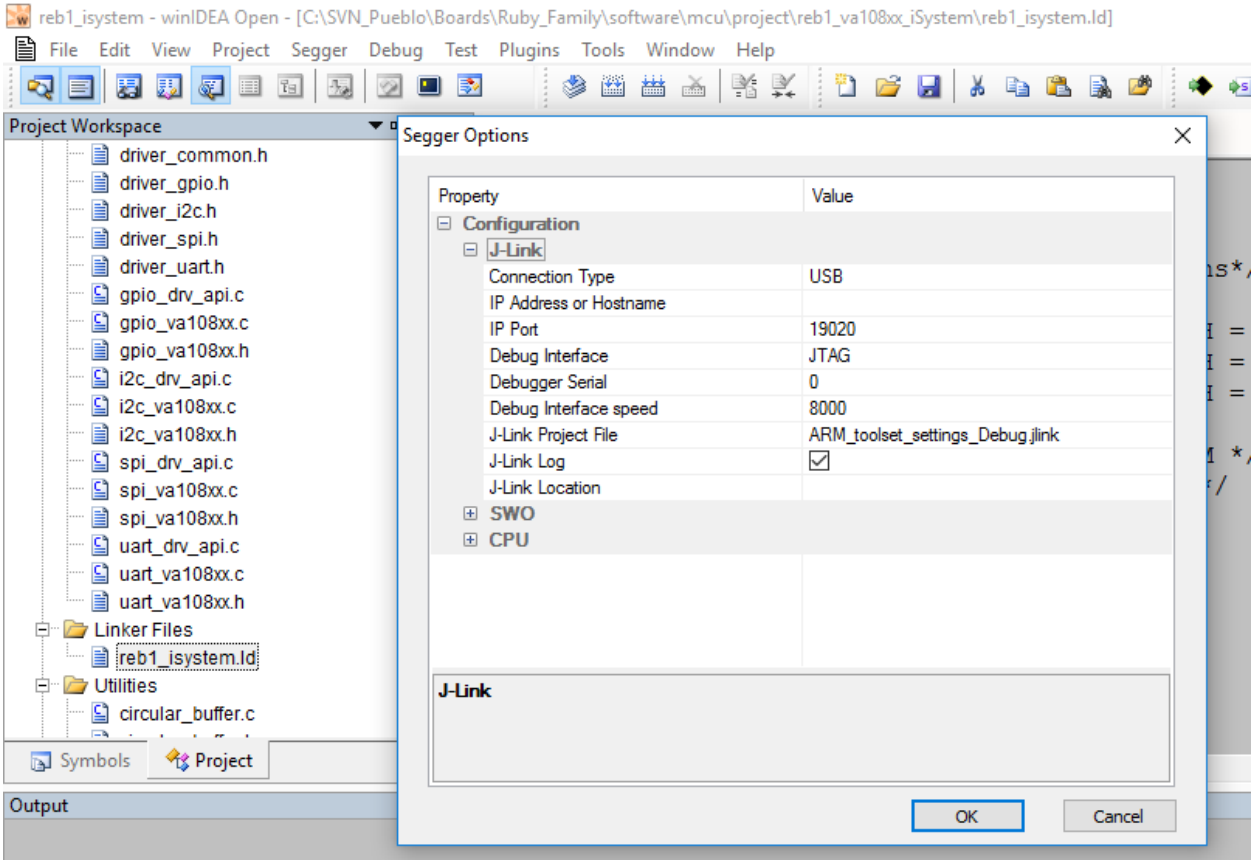
Navigate to the "reb1_va108xx_iSystem -> app -> reb1_reference -> reb1_va108xx" folder and double click "reb1_gcc_jlink.xjrf" (type=WinIDEA workspace). The winIDEA Open IDE will open with the REB1 project preloaded.

This project has the target selected to "VA108xx_reb1_Sample" which calls out the va108xx device. (Pulldown menu: Project -> Targets). The linker get information from "va108xx_ram.ld" which has memory map information.



4.3.2 Configuring options

Although the project should be ready to compile and debug on the REB1 board, it may be necessary to adjust some settings such as the debug connection. Under the **Segger** pull-down menu, selection **Options** then fill in the Debug interface field with JTAG.



4.3.3 Project Build (Rebuild)

The rebuild command will compile and link all files regardless of whether they changed. Under the **Project** pull-down, select **Rebuild.**

4.3.4 Download and Run

Under the **Debug** menu, select **Download** which directly loads the RAM memory of the device. Next, under the **Debug** menu, select **Run Control** and then **Run**. An indicator bar in the lower right corner of the display will show the state of the CPU.

4.3.5 UART output

This project directs information to a UART port with TX on PORTA[9] and RX on PORTA[8]. A virtual serial communication cable such as FTDI "TTL-232R-3V3-WE" or *SparkFun USB to RS232-*

TTL (https://www.sparkfun.com/products/12731) can be used to connect a PC to the board. PORTA[9] and PORTA[8] are available on connector J10 of the REB1 board.

4.3.6 iSystem iTAG

iSystem offers a suite of debug probe hardware. The iTAG50 probe is a solution with many features. More information can be found at http://www.isystem.com/products/hardware/cortex-debugger/itag. This product is very low cost and provides programming support for the VA108xx family of products.

PC connections must be made as shown in configuration 2 of Figure 3. Place a jumper in J21 before powering the board.

A separate project is available for this: "reb1_gcc_iTag50.xjrf". This project does not support RTT logging but it provides a debugger terminal for viewing printf outputs. There is a conditional compiler directive for selecting UART or iSystem terminal output in "reb_log.c".

```
#ifdef RTT_LOG
        SEGGER_RTT_printf(0, (const char *)log_buff) ;
#elif defined DEBUG_ISYS
          printf(const char *) log_buff)  ;
#else
          drv->Send(log_buff,  len) ;
#endif
```

If you would like to use the UART output, comment out the line beginning with #elif and the following line. When using the iSystem terminal, the debugger terminal window must be active (View -> Terminal) and the "connect" icon in the upper left corner of the terminal window be clicked.

4.3.6.1 Programming with iSystem iTAG50

After double clicking on "reb1_gcc_iTag50.xjrf", two new options will be available:

1) Under Hardware pull down menu, Flash setup and
2) A new top level menu "FLASH"

Nothing needs to be done under the top level FLASH menu. Under the Hardware -> FLASH Setup, a Devices dialog box is available. Make sure the box next to ST MA95M01 is selected.

When a project is downloaded, the file will be loaded to RAM and then the EEPROM will be programmed. A dialog box will show the progress. Pressing the RESET button on the REB1 board will force the MCU to read the EEPROM and load the on-chip RAM then begin executing

code. Alternatively, it is still possible to use the RUN, STOP and RESET buttons to control processor activity.

4.4 J-Link OB and RTT (Real Time Terminal)

To visually monitor activity in the MCU, we have incorporated code that streams information via the JTAG interface to a PC. The utility that Segger provides for this is called RTT or Real Time Terminal. Chapter 12 of the Segger J-LINK User's Manual is dedicated to explaining this function. It is included with the Segger download. This section just covers the essentials of RTT.

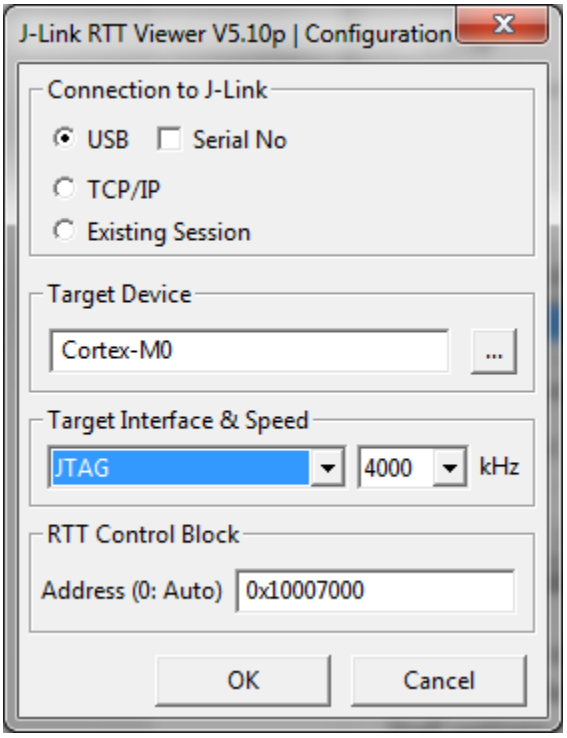
SEGGER's Real Time Terminal (RTT) is a technology for interactive user I/O in embedded applications. It combines the advantages of SWO and semihosting at very high performance. With RTT it is possible to output information from the target microcontroller as well as sending input to the application at a very high speed without affecting the target's real time behavior. No additional hardware or pin assignments are necessary for RTT to operate.

4.4.1 RTT Client application

Included in the Segger download is an application titled "J-LINK RTT Viewer". This application allows information sent from the MCU to be either displayed on a terminal or logged into a file. Alternatively, data from the keypad of the PC can be sent to the MCU.

When the application is first opened, a dialog box is shown that allows the user to configure the connection. Please set the below options and control block address. The control block is a designated piece of memory for buffering information to and from the MCU. It is configured in software as shown in the next section.

Note: An error message may be displayed stating the J-Link does not recognize the MCU. If that occurs, hit "ok" and select the generic Cortex-M0 in the subsequent dialogue box. This error message should only occur on the first attachment to the board with the MDK.



4.4.2 Embedded Software to enable RTT

Inside the REB1 demonstration program, there is a series of subroutines provided by Segger to configure the RTT, transmit and receive characters. Inside of this code, buffers are created and the RTT Control Block address is fixed at 0x10007000.

4.4.3 Starting and Stopping RTT Viewer

Starting the viewer can occur at any time during a debug session. However, after the Viewer is connected and the MCU exits a debug session, the viewer will error out since the host software is no longer active. When this occurs, it is necessary to exit the RTT Viewer and restart it.

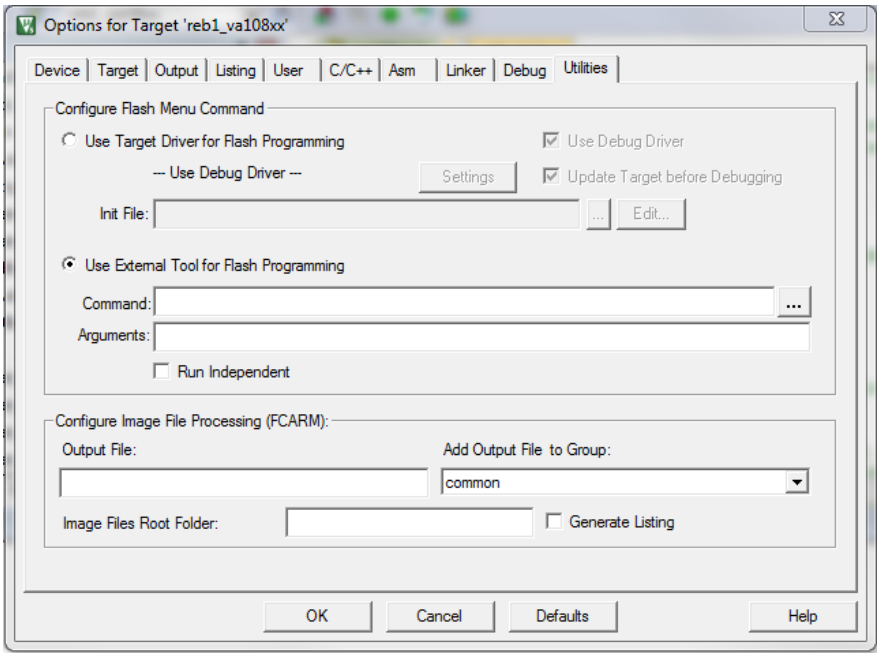
4.5 Programming procedure (Keil Specific)

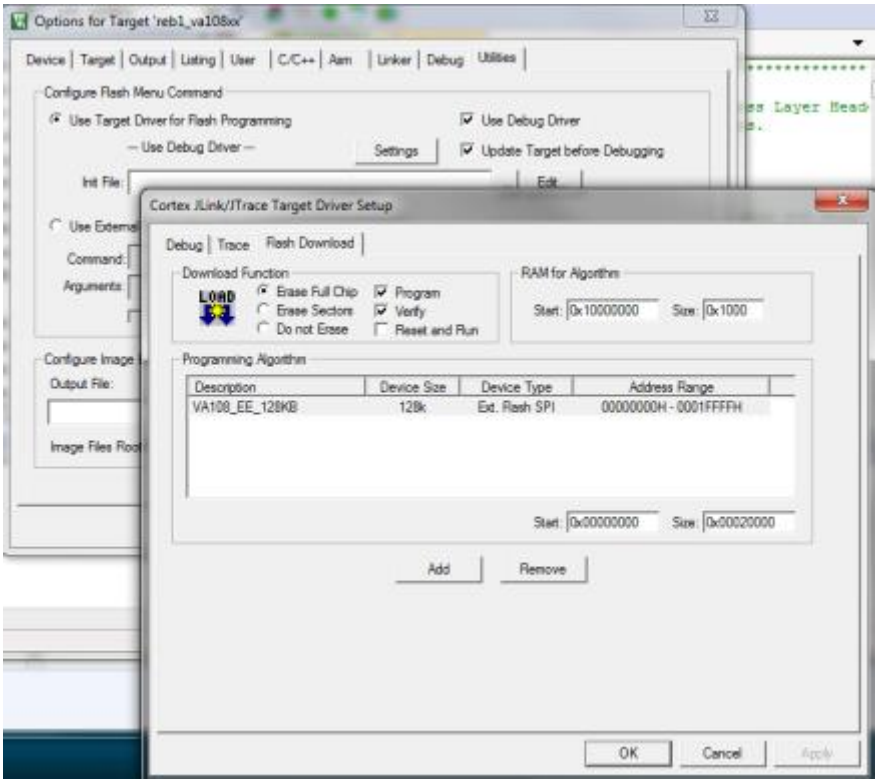
The VA108xx MCUs rely on an external SPI based memory device to boot from. The 128kbyte memory is transferred to the MCU's program RAM automatically during the MCU boot process by a hardware bootloader. The location of the code in the SPI memory device is the same as the location inside the MCU. For instance, the RESET vector information is located at address 0x0000 in both devices.
The Keil IDE provides options for programming either the flash embedded on MCUs or connected memories such as an EEPROM via the JTAG port. VORAGO has provided a specific programming algorithm for the EEPROM device used on REB1. Programming the full 128kbytes

of EEPROM takes only a few seconds. Support for Cypress FRAM, TT Semi Flash and Everspin MRAM is also available.

4.5.1 Utility Options

Before programming the EEPROM that boots the VA108xx, the Utilities options menu must be set up. First click on the **Use Target Driver for Flash Programming** button. Then click on the **Settings** button. Under the programming Algorithm section, click on the **Add** button, then select the "VA108_EE_128k" in the list. Click "Ok" and "Ok" to exit the options menu.





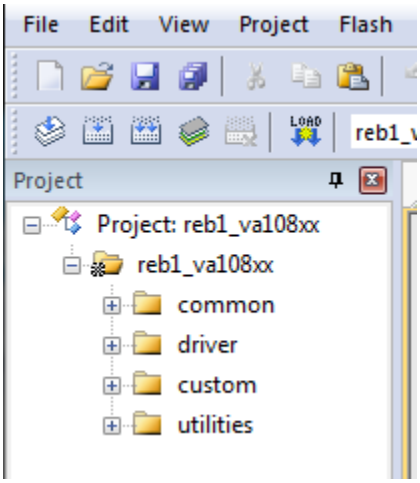
Note: When not programming the EEPROM, remove the VA108_EE_128KB file from the programming algorithm list and click the **Use external flash programming** radial button. Failure to do this will result in unpredictable debugger operation.

5 Software Development Kit

VORAGO provides a good starting point for end application development with the REB1 SDK (software development kit). This section gives a brief tour of the kit.

5.1 Project organization

Inside each IDE, a project can be organized in groups to facilitate re-use of proven components. The demonstration program provided organizes the various .c and .h files in four different groups as shown below. These groups are not file folders. Each group has files that can reside in unique folders on a PC.



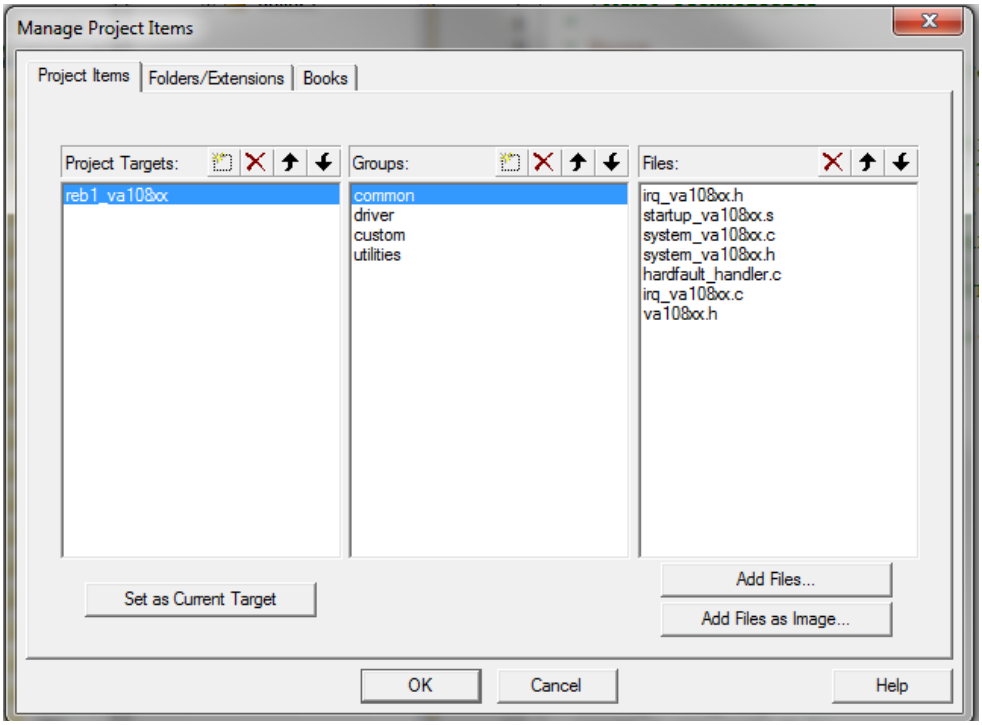


Figure 4 - Keil IDE project item management folder

The "common" group has files specific the MCU. The "driver" group has files for using the peripheral modules of the VA108xx MCU. The "custom" group has files unique for this project including the main.c file. The "utilities" group has Segger RTT support files and some redirect functions that allow peripherals with multiple instances to use the same code.

5.2 CMSIS compatible driver

ARM has created the Cortex-M Software Interface Standard (CMSIS) to provide a basis for structuring code such that it can be ported from one ARM based MCU to another. Details on

this standard can be found at: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php.

A separate set of documentation in Doxygen format is available with the REB1 download in the documentation folder. Navigate to the "index.html" file and double click on it to access the Doxygen information for the board and software.

5.3 Preprocessor directives

For bigger projects, the use of preprocessor directives can allow different build options. The below code snippet shows the "ifdef" preprocessor directive. If GPIO_DRIVER was defined, the first portion of the snippet would be executed. If GPIO_DRIVER was not defined, the "else" portion would be executed. FYI: The demonstration code does not utilize the GPIO_DRIVER.

```
58    /* Configure PA[10]. This GPIO will continously blink led D2. */
59  #ifdef GPIO_DRIVER
60    PA10->Initialize(NULL);
61    PA10->Control(VOR_GPIO_CONTROL_MASK, VOR_GPIO_CTRL_ENABLE);
62    PA10->Control(VOR_GPIO_CONTROL_DIR, VOR_GPIO_DIR_OUTPUT);
63    PA10->Write(1);
64  #else
65    VOR_GPIO->BANK[0].DIR |= (1 << PORTA_10_D2);
66    VOR_GPIO->BANK[0].DATAMASK |= (1 << PORTA_10_D2);
67    VOR_GPIO->BANK[0].DATAOUT &= ~(1 << PORTA_10_D2);
68  #endif
```

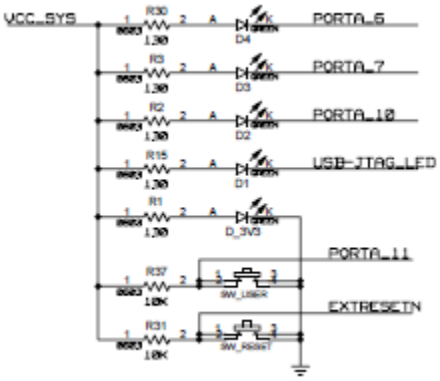
6 Lab exercises

The following labs will use the SDK as the base software and will provide step by step instructions for adding new functionality to some of the most common functional blocks on the device such as SPI and GPIO. Manual input of text is kept to a minimum.

6.1 Lab 1 – Toggling an output pin to blink an LED

The demonstration project already has LED D2 blinking every 100 msec. This lab will blink LED D4 every 400 msec.

First the schematic should be referenced to determine which port pin is connected to LED D4. PORTA6 controls D4. To turn on the LED, the pin must be driven low. We will need to setup the pin to be an output and then toggle between a one and zero inside a 100 msec interval routine.



Second, the main routine (inside file reb_main.c) should be opened and some code added. If not already open, please double click on the demonstration project, ../software/mcu/project/reb1_va108xx. Navigate to a section of the code around line 58 that already sets up the port pin for driving LED2.

```
58    /* Configure PA[10]. This GPIO will continously blink led D2. */
59  #ifdef GPIO_DRIVER
60    PA10->Initialize(NULL);
61    PA10->Control(VOR_GPIO_CONTROL_MASK, VOR_GPIO_CTRL_ENABLE);
62    PA10->Control(VOR_GPIO_CONTROL_DIR, VOR_GPIO_DIR_OUTPUT);
63    PA10->Write(1);
64  #else
65    VOR_GPIO->BANK[0].DIR |= (1 << PORTA_10_D2);
66    VOR_GPIO->BANK[0].DATAMASK |= (1 << PORTA_10_D2);
67    VOR_GPIO->BANK[0].DATAOUT &= ~(1 << PORTA_10_D2);
```

Since we are not using the GPIO_DRIVER library, the "else" section will be compiled. Modify the lines that setup the direction, datamask and dataout registers to include PORTA6. It should look something similar to:

```
VOR_GPIO->BANK[0].DIR |= (1 << PORTA_10_D2) | (1 << PORTA_6_D4);
VOR_GPIO->BANK[0].DATAMASK |= (1 << PORTA_10_D2) | (1 << PORTA_6_D4);
VOR_GPIO->BANK[0].DATAOUT &= ~((1 << PORTA_10_D2| (1 << PORTA_6_D4)));
```

Next we need to toggle the pin on 400 msec intervals. Navigate to code near line 120 in reb_main.c. This is a subroutine that is entered every 100 msec. The time base is set by a timer interrupt. We will need to count to 4 before toggling the port pin. A variable must be created for this counting purpose. Place the following code at the top of the reb_main.c routine with the other variable declarations:

```
uint8_t cnt_400msec = 0 ;
```

Now, please enter the following code in green to count to 4 and toggle the pin.

```
if( VOR_TIM0_MSecExpired() ) {
    cnt_100msec ++ ;
    if (cnt_100msec > 3)
```

```
{
        cnt_100msec = 0 ;   //  reset counter
        VOR_GPIO->BANK[0].TOGOUT |= (1 << PORTA_6_D4);
}
```

Compile, download and run the program.

6.2 LAB2 - Advanced input pin filtering and debounce of switch input.

The VA108xx comes with advanced filtering schemes to eliminate glitches being read and wasting CPU time filtering them out. See IOCONFIG peripheral for more detailed information. For this example, we will enable a pullup resistor in the port pin and have logic outside the CPU sample the pin 5 times before signaling the state of the pin has changed.

Step 1: Set the sample period

We will use filter timer 1 for this purpose. By setting the clock divider to 100, we are setting the sampling period to 100 CPU cycles.

```
VOR_SYSCONFIG->IOCONFIG_CLKDIV1 |= 100;
```

Step 2: Configure the IO pin

Set the pin configuration register for the following:

- Filter type (FLTTYPE) = sample 5 times
- Filter clock source (FLTCLK) = filter clock 1
- Pull level (PLEVEL) = pull up which is 0
- Pull Enable (PEN) = 1

```
VOR_IOCONFIG->PORTA[0] |= (4 << IOCONFIG_PORTA_FLTTYPE_Pos) | (1 << IOCONFIG_PORTA_FLTCLK_Pos) |
        (1 << IOCONFIG_PORTA_PEN_Pos);
```

Step 3: Read the port pin

The pin can either be assigned an interrupt, can be polled directly or the IRQ_EVT bit can be polled. The IRQ_EVT will indicate whether an event has occurred on that pin since the last time it was cleared.

```
X = VOR_GPIO->BANK[0] -> DATAIN  ;
```

7 Commonly asked questions

1. Code runs but no activity seen on the port pin. What is going on?
    a. Two items to check are:
        i. Make sure that the peripheral clock is enabled for all the active peripherals. See PERIPHERAL_CLK_ENABLE Register in the programmers

guide. The IOCONFIG peripheral is often an overlooked peripheral that must be enabled if using a peripheral module to control a port pin.

ii. The functional selection bits must be set to allow a peripheral to control a port pin. See IO Configuration Register

2. The debugger downloaded code but when instructed to run, the code does not execute. What might be causing this?
    a. The Keil IDE must have the flash utility totally disabled before downloading code directly to RAM. Open the Utility window, deselect the programming file and select "Use External" programmer. From that point on, the debugger should download code correctly.
    b. If the hardware RESET button is pressed on the board, the device will reload program RAM with information from the EEPROM which overwrites what the debugger placed in RAM. If this is the case, reload the code via the IDE.

3. My code runs from the debugger, but does not run or runs an older program when powered up on its own.
    a. Keil is targeting only volatile program RAM when debugging and not using the programming algorithm to target the SPI EEPROM. All rebooting forces the VA108xx to reload its program RAM from EEPROM. Refer to section 3.7.1 for programming to EPROM.

4. What is the minimum number of instructions to send information out a UART?
    a. The below code can be run to output a 9600 baud message out UARTA on PORTA[9].

```
#define FUNCSEL2 2   //  used for assigning UARTA to PORTA[9]

VOR_SYSCONFIG->PERIPHERAL_CLK_ENABLE = (  CLK_ENABLE_UARTA |CLK_ENABLE_IOCONFIG   );  // enable peripheral clocks

VOR_IOCONFIG->PORTA[9] |= (FUNCSEL2 << IOCONFIG_PORTA_FUNSEL_Pos);  // assign UARTA to PORTA[9]

VOR_UART->BANK[0].CTRL = 0x30 ;   //  this is default value, no parity, 8-bits
VOR_UART->BANK[0].CLKSCALE = 0x5161;  //  baud rate = 50Mhz /
VOR_UART->BANK[0].ENABLE = 2  ;   // only enable TX = bit position 1.
VOR_UART->BANK[0].DATA = 0x5A ;  //  send single byte out.
```

8 Other resources for VA108x0 code

Vorago application notes: http://www.voragotech.com/resources

9 Revision history

1.0 April 2016

2.0 November 2016

2.1 Includes sections for IAR and iSystem

2.2 Corrected connector definition table.

3.0 February 2017

3.1 Updated iSystem section for latest release.

3.2 Added text to show how to switch from RTT to UART logging

3.3 Removed section with workaround for SVD formatting error message. (No longer needed)

3.4 Added two new Frequently asked questions in section 7.