

AN1321 – Creating a New VA416xx Project in VS Code IDE

August 2024 Version 0.1

VA416xx

Abstract

This document is intended to provide instructions on how to add your own user firmware project to the provided VS Code / GCC workspace. Support for this development environment is included in the VA416xx BSP version 2.00 and up. Because of the nature of open source tools, adding a project to the workspace involves editing JSON and other text files, and is not as straightforward as with the licensed tools like Keil or IAR. This step by step set of instructions is based around making a copy of the ‘blinky’ demo project, which can then be used as a basis for an end user application.

Table of Contents

1	Prerequisites	1
2	Creating the project.....	1
3	Building and running the new project	7
4	Conclusion	10

1 Prerequisites

To use this application note, first follow the PEB1 user’s guide to setup Segger J-link, VS Code, VS Code extensions, GCC, and Make / CMake.

2 Creating the project

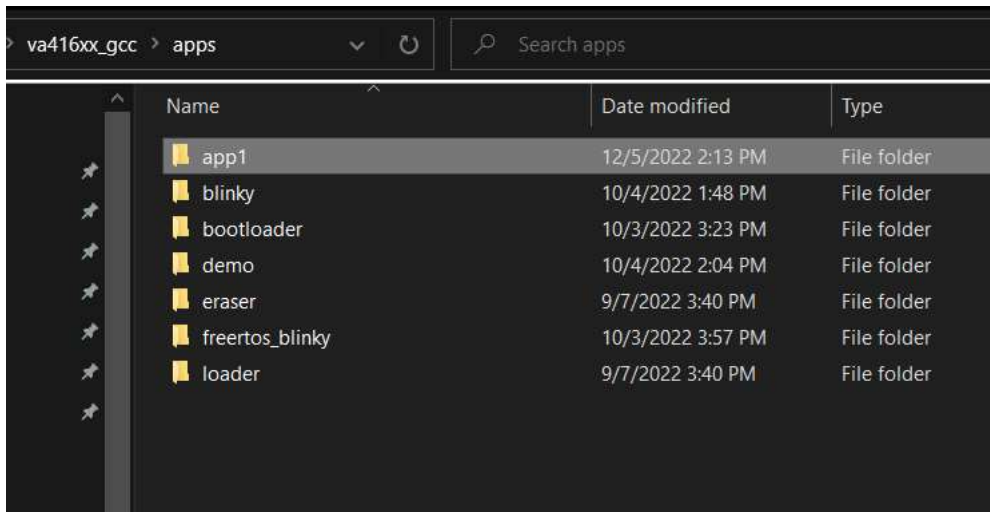
2.1 Choose a name for the project and create required folders

The first step is to decide on a name for the project. It will often be typed into the terminal, so it is best to keep it short. Here, we will call the new project ‘app1’.

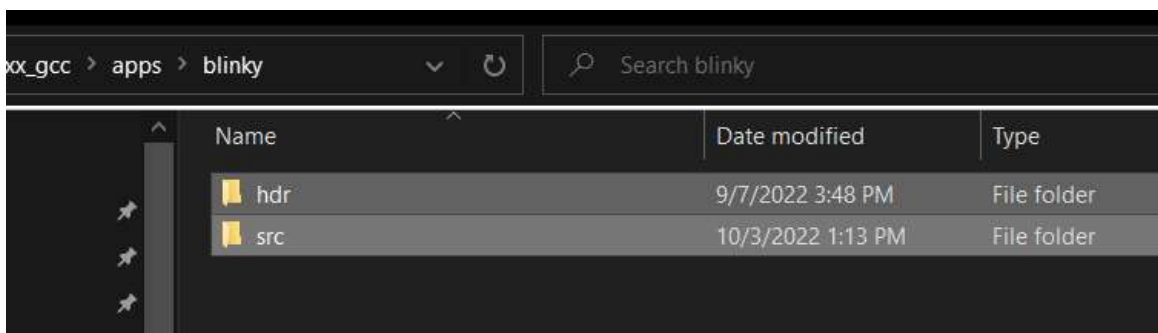
2.1.1 Create directories

In the VS code workspace folder (here it is called ‘va416xx_gcc’), under ‘apps’, create a new folder and name it ‘app1’.

AN1321 – Creating a New VA416xx Project in VS Code IDE



Navigate over to the 'blinky' app, and copy the 'hdr' and 'src' folders from blinky project to the 'app1' folder.



Navigate into the 'src' folder just created in 'app1'. Open the file 'CMakeLists.txt' in a text editor. Change 'BLINKY_SRC' to 'APP1_SRC'. Save and close the file.

```
*CMakeLists.txt - Notepad
File Edit Format View Help
set(src "APP1_SRC")

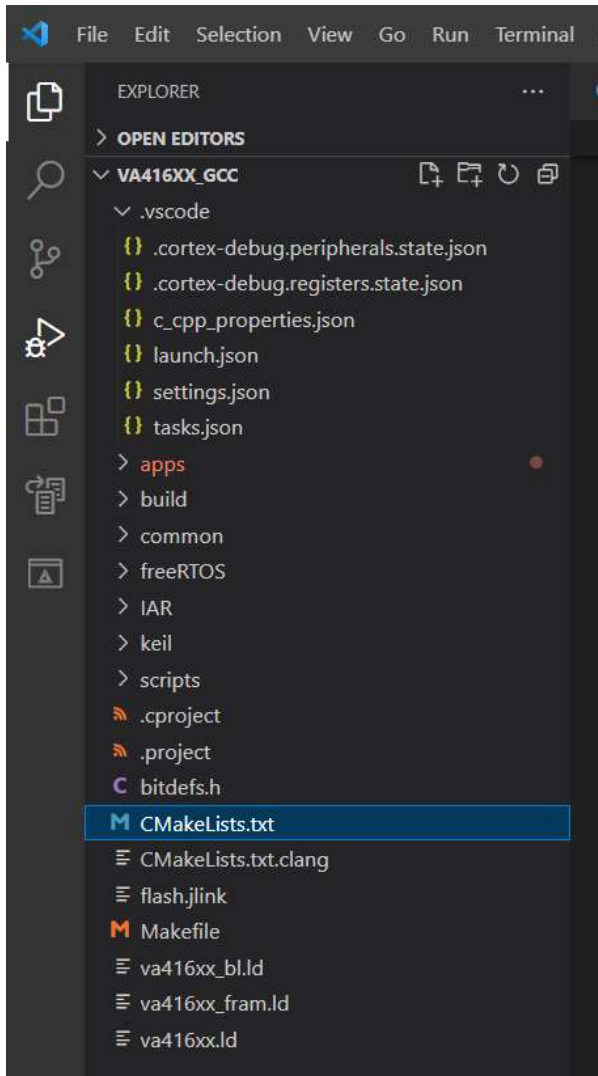
file(GLOB FILES *.c startup_va416xx.s)
list(APPEND "${src}" "${FILES}")
set("${src}" "${${src}} PARENT_SCOPE)
```

2.2 Setup the makefiles

2.2.1 Edits to top level CMakeLists.txt

In VS code, open \$WorkspaceDirectory/CMakeLists.txt.

AN1321 – Creating a New VA416xx Project in VS Code IDE



Find the section with the comment '# Apps source paths', and add a line for the app1 project:

```
108
109 # Apps Source paths
110 add_subdirectory(apps/bootloader/src)
111 add_subdirectory(apps/loader)
112 add_subdirectory(apps/eraser)
113 add_subdirectory(apps/blinky/src)
114 add_subdirectory(apps/freertos_blinky/src)
115 add_subdirectory(apps/demo/src)
116 add_subdirectory(apps/app1/src)
```

At the end of the file, add a 'build_target' line for the new application. If the new app is a FreeRTOS project, call 'build_freertos_target' instead of 'build_target'. Save and close the file.

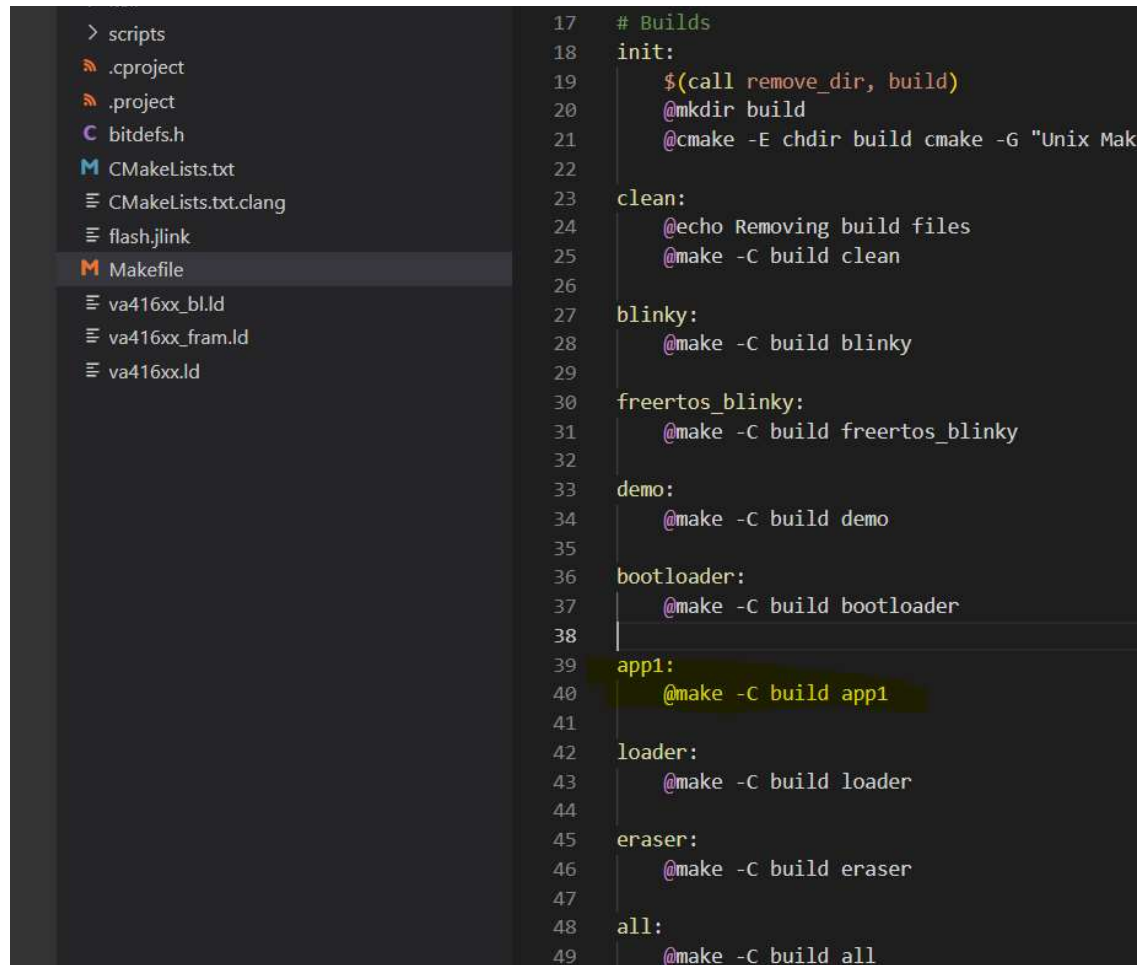
```

167
168 build_target(bootloader BOOTLOADER_SRC)
169
170 build_loader(loader LOADER_SRC)
171 build_loader(eraser ERASER_SRC)
172
173 build_target(blinky BLINKY_SRC)
174 build_freertos_target(freertos_blinky FREERTOS_BLINKY_SRC)
175 build_target(demo DEMO_SRC)
176 build_target(app1 APP1_SRC)
177
178

```

2.2.2 Edits to top level Makefile

Open `$WorkspaceDirectory/Makefile`. In the '# Builds' section, add a build command for the new app (lines 39 and 40 below). Then save and close the file.



```

17 # Builds
18 init:
19     $(call remove_dir, build)
20     @mkdir build
21     @cmake -E chdir build cmake -G "Unix Makefiles"
22
23 clean:
24     @echo Removing build files
25     @make -C build clean
26
27 blinky:
28     @make -C build blinky
29
30 freertos_blinky:
31     @make -C build freertos_blinky
32
33 demo:
34     @make -C build demo
35
36 bootloader:
37     @make -C build bootloader
38
39 app1:
40     @make -C build app1
41
42 loader:
43     @make -C build loader
44
45 eraser:
46     @make -C build eraser
47
48 all:
49     @make -C build all

```

2.3 Edit the VS Code JSON files

Some VS Code configuration files will need to be edited to add the necessary information about the new project. This will tell VS Code how to run/debug the project, for example, and provide include directory information to the IntelliSense configuration.

2.3.1 Editing launch.json

In VS Code, in the 'Run' dropdown menu, choose 'open configurations'. This will open launch.json. Alternatively, it can be opened from the explorer sidebar. This file contains the run/debug information, a 'run in SRAM' and 'run in FRAM' configuration will need to be added for the new project.

First, select and copy the SRAM and FRAM configuration for the blinky project:

```
.vscode > {} launch.json > JSON Language Features > [ ] configurations
1  {
2      "version": "0.2.0",
3      "configurations": [
4
5          {
6              "name": "JLink: blinky run in SRAM",
7              "cwd": "${workspaceFolder}",
8              "request": "launch",
9              "type": "cortex-debug",
10             "serverType": "jlink",
11             "executable": "${workspaceFolder}/build/blinky.elf",
12             "interface": "swd",
13             "device": "Cortex-M4",
14             "runToEntryPoint": "main",
15             "svdFile": "scripts/va416xx.svd",
16             "overrideLaunchCommands": [
17                 "monitor reset 0",
18                 "monitor halt",
19                 "set *((unsigned int *) 0x40010010) = 0x00000001",
20                 "load ./build/blinky.elf",
21                 "set *((unsigned int *) 0x40010010) = 0x00000000",
22             ],
23         },
24         {
25             "name": "JLink: blinky run in FRAM",
26             "cwd": "${workspaceFolder}",
27             "request": "launch",
28             "type": "cortex-debug",
29             "serverType": "jlink",
30             "executable": "${workspaceFolder}/build/blinky.elf",
31             "interface": "swd",
32             "device": "Cortex-M4",
33             "runToEntryPoint": "main",
34             "svdFile": "scripts/va416xx.svd",
35             "overrideLaunchCommands": [
36                 "monitor reset 0",
37                 "monitor halt",
38                 "set *((unsigned int *) 0x40010010) = 0x00000001",
39                 "load ./build/blinky.elf",
40                 "set *((unsigned int *) 0x40010010) = 0x00000000",
41                 "load ./build/loader.elf",
42             ],
43         },
44     ],
45     {
46         "name": "JLink: freertos blinky run in SRAM",
```

Paste the copy of this configuration just below the blinky config. In the “name” field, replace ‘blinky’ with ‘app1’. Replace occurrences of ‘blinky.elf’ with ‘app1.elf’. Save and close the file.

```

41         "load ./build/loader.elf",
42     ],
43 },
44 {
45     "name"           : "JLink: app1 run in SRAM",
46     "cwd"            : "${workspaceFolder}",
47     "request"        : "launch",
48     "type"           : "cortex-debug",
49     "servertype"     : "jlink",
50     "executable"     : "${workspaceFolder}/build/app1.elf",
51     "interface"      : "swd",
52     "device"         : "Cortex-M4",
53     "runToEntryPoint": "main",
54     "svdFile"        : "scripts/va416xx.svd",
55     "overrideLaunchCommands" : [
56         "monitor reset 0",
57         "monitor halt",
58         "set *((unsigned int *) 0x40010010) = 0x00000001",
59         "load ./build/app1.elf",
60         "set *((unsigned int *) 0x40010010) = 0x00000000",
61     ],
62 },
63 {
64     "name"           : "JLink: app1 run in FRAM",
65     "cwd"            : "${workspaceFolder}",
66     "request"        : "launch",
67     "type"           : "cortex-debug",
68     "servertype"     : "jlink",
69     "executable"     : "${workspaceFolder}/build/app1.elf",
70     "interface"      : "swd",
71     "device"         : "Cortex-M4",
72     "runToEntryPoint": "main",
73     "svdFile"        : "scripts/va416xx.svd",
74     "overrideLaunchCommands" : [
75         "monitor reset 0",
76         "monitor halt",
77         "set *((unsigned int *) 0x40010010) = 0x00000001",
78         "load ./build/app1.elf",
79         "set *((unsigned int *) 0x40010010) = 0x00000000",
80         "load ./build/loader.elf"
81     ],
82 },
83 {

```

2.3.2 Editing c_cpp_properties.json

This file defines the IntelliSense configuration. A new configuration for the ‘app1’ project will be created. Open ‘c_cpp_properties.json’. Copy the ‘blinky’ configuration, and paste it below blinky.

AN1321 – Creating a New VA416xx Project in VS Code IDE

```
.vscode > {} c_cpp_properties.json > [] configurations > {} 0
1 {
2   "configurations": [
3     {
4       "name": "blink",
5       "includePath": [
6         "${workspaceFolder}/apps/blink/**",
7         "${workspaceFolder}/common/**"
8       ],
9       "defines": [],
10      "compilerPath": "C:\\Program Files (x86)\\GNU Arm Embedded Toolchain\\10.2021.10\\bin\\arm-none-eabi-gcc.exe",
11      "cStandard": "gnu17",
12      "cppStandard": "gnu++17",
13      "intelliSenseMode": "windows-gcc-arm",
14      "configurationProvider": "ms-vscode.cmake-tools"
15    },
16    {
17      "name": "freertos_blink",
```

Edit the new configuration. Change the “name” to “app1”, and the “includePath” to include “\${workspaceFolder}/apps/app1/**” instead of “\${workspaceFolder}/apps/blink/**”

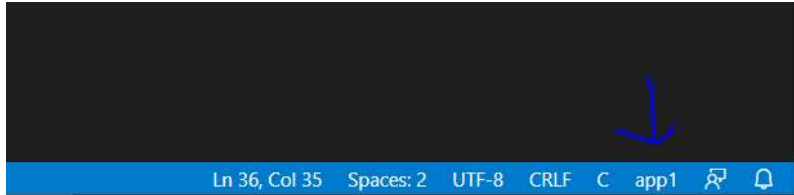
```
1 {
2   "configurations": [
3     {
4       "name": "blink",
5       "includePath": [
6         "${workspaceFolder}/apps/blink/**",
7         "${workspaceFolder}/common/**"
8       ],
9       "defines": [],
10      "compilerPath": "C:\\Program Files (x86)\\GNU Arm Embedded Toolchain\\
11      "cStandard": "gnu17",
12      "cppStandard": "gnu++17",
13      "intelliSenseMode": "windows-gcc-arm",
14      "configurationProvider": "ms-vscode.cmake-tools"
15    },
16    {
17      "name": "app1",
18      "includePath": [
19        "${workspaceFolder}/apps/app1/**",
20        "${workspaceFolder}/common/**"
21      ],
22      "defines": [],
23      "compilerPath": "C:\\Program Files (x86)\\GNU Arm Embedded Toolchain\\
24      "cStandard": "gnu17",
25      "cppStandard": "gnu++17",
26      "intelliSenseMode": "windows-gcc-arm",
27      "configurationProvider": "ms-vscode.cmake-tools"
28    },
29  ],
30 }
```

Save and close the JSON file.

3 Building and running the new project

3.1 Setting the new C/C++ configuration

Open apps/app1/src/main.c. In the lower right corner of the VS code window, select the 'app1' configuration. This will tell VS Code where the include paths are for this application, so the autofill and commands like 'go to definition' when right clicking on a function or variable name will work correctly.



3.2 Building the new project

Since the makefiles were edited, a “make init” must be performed. Run this command in the terminal:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\proj\va416xx_gcc> make init
-- The C compiler identification is GNU 10.3.1
-- The CXX compiler identification is GNU 10.3.1
-- The ASM compiler identification is GNU
-- Found assembler: C:/Program Files (x86)/GNU Arm Embedded Toolchain/10 2021.10/bin/arm-none-eabi-gcc.exe
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files (x86)/GNU Arm Embedded Toolchain/10 2021.10/bin/arm-none-eabi-gcc.exe - skip
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/GNU Arm Embedded Toolchain/10 2021.10/bin/arm-none-eabi-g++.exe - sk
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/proj/va416xx_gcc/build
PS C:\proj\va416xx_gcc> 
```

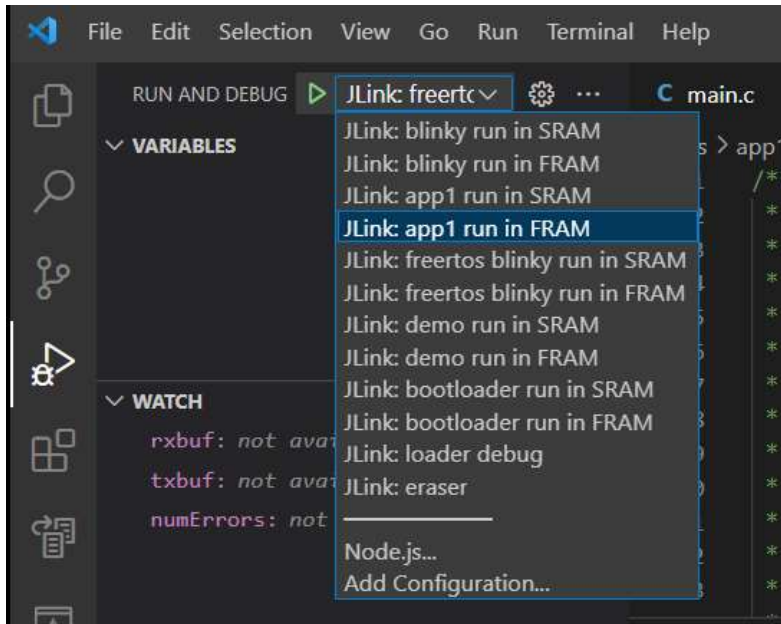
Next, run “make app1”:

AN1321 – Creating a New VA416xx Project in VS Code IDE

```
-- Build files have been written to: C:/proj/va416xx_gcc/build
PS C:\proj\va416xx_gcc> make app1
make[1]: Entering directory `C:/proj/va416xx_gcc/build'
make[2]: Entering directory `C:/proj/va416xx_gcc/build'
make[3]: Entering directory `C:/proj/va416xx_gcc/build'
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
[ 5%] Building C object CMakeFiles/eraser.dir/apps/eraser/eraser.c.o
[ 5%] Linking C executable eraser.elf
Memory region      Used Size  Region Size  %age Used
      FLASH:         0 GB      256 KB      0.00%
        RAM:        224 B        32 KB      0.68%
      CCMRAM:         0 GB        32 KB      0.00%
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
[ 5%] Built target eraser
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
[ 10%] Building C object CMakeFiles/loader.dir/apps/loader/loader.c.o
[ 15%] Linking C executable loader.elf
Memory region      Used Size  Region Size  %age Used
      FLASH:         0 GB      256 KB      0.00%
        RAM:        252 B        32 KB      0.77%
      CCMRAM:         0 GB        32 KB      0.00%
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
[ 15%] Built target loader
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
Scanning dependencies of target app1
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
make[4]: Entering directory `C:/proj/va416xx_gcc/build'
[ 15%] Building C object CMakeFiles/app1.dir/apps/app1/src/board.c.o
[ 20%] Building C object CMakeFiles/app1.dir/apps/app1/src/hardFault_handler.c.o
[ 20%] Building C object CMakeFiles/app1.dir/apps/app1/src/main.c.o
[ 25%] Building ASM object CMakeFiles/app1.dir/apps/app1/src/startup_va416xx.s.o
[ 25%] Building C object CMakeFiles/app1.dir/apps/app1/src/uart.c.o
[ 30%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_debug.c.o
[ 30%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal.c.o
[ 35%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_adc.c.o
[ 40%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_adc_swcal.c.o
[ 40%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_canbus.c.o
[ 45%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_clkgen.c.o
[ 45%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_dac.c.o
[ 50%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_dma.c.o
[ 50%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_ethernet.c.o
[ 55%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_i2c.c.o
[ 60%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_ioconfig.c.o
[ 60%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_irqrouter.c.o
[ 65%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_spi.c.o
[ 65%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_spw.c.o
[ 70%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_timer.c.o
[ 70%] Building C object CMakeFiles/app1.dir/common/drivers/src/va416xx_hal_uart.c.o
[ 75%] Building C object CMakeFiles/app1.dir/common/mcu/src/system_va416xx.c.o
[ 75%] Building C object CMakeFiles/app1.dir/common/utils/src/circular_buffer.c.o
[ 80%] Building C object CMakeFiles/app1.dir/common/utils/src/dac_sine.c.o
[ 85%] Building C object CMakeFiles/app1.dir/common/utils/src/segger_rtt.c.o
[ 85%] Building C object CMakeFiles/app1.dir/common/utils/src/segger_rtt_printf.c.o
[ 90%] Building C object CMakeFiles/app1.dir/common/Utils/src/spi_fram.c.o
[ 90%] Building C object CMakeFiles/app1.dir/common/BSP/evk/src/evk_board.c.o
[ 95%] Building C object CMakeFiles/app1.dir/common/BSP/evk/src/i2c_adxl343.c.o
[ 95%] Building C object CMakeFiles/app1.dir/common/BSP/evk/src/i2c_lis2de12.c.o
[100%] Linking C executable app1.elf
Memory region      Used Size  Region Size  %age Used
      FLASH:    12120 B      256 KB      4.62%
        RAM:    4256 B        32 KB     12.99%
      CCMRAM:         0 GB        32 KB      0.00%
   text  data  bss  dec  hex filename
  11936   184   4080  16200  3f48 app1.elf
make[4]: Leaving directory `C:/proj/va416xx_gcc/build'
[100%] Built target app1
make[3]: Leaving directory `C:/proj/va416xx_gcc/build'
make[2]: Leaving directory `C:/proj/va416xx_gcc/build'
make[1]: Leaving directory `C:/proj/va416xx_gcc/build'
PS C:\proj\va416xx_gcc> █
```

3.3 Running the new project

On the left-hand side of the IDE, click on the 'Run and debug' tab. In the Run and Debug dropdown menu, choose the 'JLink: app1 run in FRAM' configuration:



Next, click the green triangle to start the debug session. If in 'run in FRAM' mode, the code will be persistent and loaded into NVM on the board. If choosing a 'run in SRAM' configuration, the code will only be loaded into RAM and if the processor resets, the code stored in NVM will be booted instead after the reset. It is recommended to run from FRAM unless there is a specific reason to preserve the NVM contents.

4 Conclusion

By following this guide, a copy of the 'blinky' project has been made in an existing workspace. This new project can be used as a basis for the user application by modifying / adding source files to the application folder. For a different application name, follow the above instructions but replace 'app1' with the desired project name. For a new freeRTOS project, follow the above steps but copy the freeRTOS_blinky project instead.

Revision log:

August 13, 2024 – Initial template created (V0.1)