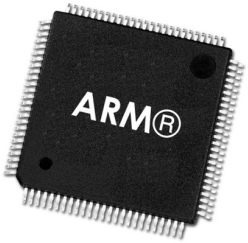


RENESAS RA8 MCU High Performance User Guide

Application Note
Renesas RA Family
High Performance with
RA8 MCU using Arm®
CortexM85 core with Helium™

Contents

Introduction

This application note describes the creation of applications with improved performance with Renesas RA8 MCUs using Cortex-M85 (CM85) core with Helium™. It is intended to highlight the performance advantages of the Arm® Cortex-M85 core, including low latency operation. Helium, Arm's M-Profile vector extension with integer and floating-point support enables advanced Digital Signal Processing (DSP), Machine Learning (ML) capabilities and helps accelerate compute-intensive applications such as endpoint Artificial Intelligence (AI), ML.
This application note walks you through all the steps necessary to achieve higher performance, including:

- Application overview
- Application highlights

- Tool configuration
- Application confirmation

Required Resources
Development Tools and Software

- IAR Embedded Workbench (IAR EWARM) version 9.40.1.63915 or later
- Renesas Flexible Software Package (FSP) v5.0.0 or later.

Hardware

- Renesas EK-RA8M1 kit (RA8M1 MCU Group)

Reference Manuals

- RA Flexible Software Package Documentation Release v5.0.0
- Renesas RA8M1 Group User's Manual Rev.1.0
- EK-RA8M1-v1.0 Schematics

Application Overview

The application projects accompanying this document showcase the performance advantages of the Renesas RA8 MCU with CM85 core. Helium intrinsics and Arm® CMSIS DSP Library functions are benchmarked to highlight the improvements versus the scalar version of these intrinsics.
The applications also utilize Tightly Coupled Memory (TCM) and cache together with Helium for further performance improvement.

Arm® Cortex® -M85 Core and Helium™ Technology

Arm® Helium™ technology is the M-profile Vector Extension (MVE) for the Arm Cortex-M processor series. It is part of the Arm v8.1-M architecture and enables developers to realize a performance uplift for DSP and ML applications. Helium™ technology provides optimized performance using Single Instruction Multiple Data (SIMD) to perform the same operation simultaneously on multiple data. There are two variants of MVE, the integer and floating-point variant:

- MVE-I operates on 32-bit, 16-bit, and 8-bit data types, including Q7, Q15, and Q31.
- MVE-F operates on half-precision and single-precision floating-point values.
  MVE operations are divided orthogonally in two ways, lanes, and beats.
- **Lanes**
  Lane is a portion of a vector register or operation. The data that is put into a lane is referred to as an element.
  Multiple lanes can be executed per beat. There are four beats per vector instruction. The permitted lane widths,
  and lane operations per beat, are: – For a 64-bit lane size, a beat performs half of the lane operation.
  – For a 32-bit lane size, a beat performs a one lane operation.
  – For a 16-bit lane size, a beat performs a two-lane operation.
  – For an 8-bit lane size, a beat performs four lane operations.
- **Beats**
  Beat is a quarter of an MVE vector operation. Because the vector length is 128 bits, one beat of a vector add

instruction equates to computing 32 bits of result data. This is independent of lane width. For example, if a lane width is 8 bits, then a single beat of a vector add instruction would perform four 8-bit additions. The number of beats for each tick describes how much of the architectural state is updated for each architecture tick in the common case. Systems are classified by:

– In a single-beat system, one beat might occur for each tick.

– In a dual-beat system, two beats might occur for each tick.

– In a quad-beat system, four beats might occur for each tick.

Cortex® -M85 implements a dual-beat system, and it supports overlapping up to two beat-wise MVE instructions at any time so that an MVE instruction can be issued after another MVE instruction without additional stall . Refer to Arm® Cortex® -M85 Processor Devices for more information.

2.1 Arm® Cortex® -M85 core

Main features of Arm® Cortex® -M85 core in Renesas RA8 MCU are as follows.

- Maximum operating frequency: up to 480 MHz
- Arm® Cortex® -M85 core
  – Revision: (r0p2-00rel0)
  – Armv8.1-M architecture profile
  – Armv8-M Security Extension
  – Floating Point Unit (FPU) compliant with the ANSI/IEEE Std 754-2008 Scalar half, single, and double-precision floating-point operation
  – M-profile Vector Extension (MVE) Integer, half-precision, and single-precision floating-point MVE (MVE-F)
  – – Helium™ technology is M-profile Vector Extension (MVE)
- Arm® Memory Protection Unit (Arm MPU)
  – – Protected Memory System Architecture (PMSAv8)
  — Secure MPU (MPU_S): 8 regions
  — Non-secure MPU (MPU_NS): 8 regions
- SysTick timer
  — Embeds two Systick timers: Secure instance (SysTick_S) and Non-secure instance (SysTick_NS)
  — Driven by CPUCLK or SYSTICKCLK (MOCO/8).
- CoreSight™ ETM-M85

Figure 1 shows the block diagram of Arm® Cortex® -M85 core.

Figure 1 shows the block diagram of Arm® Cortex®-M85 core.

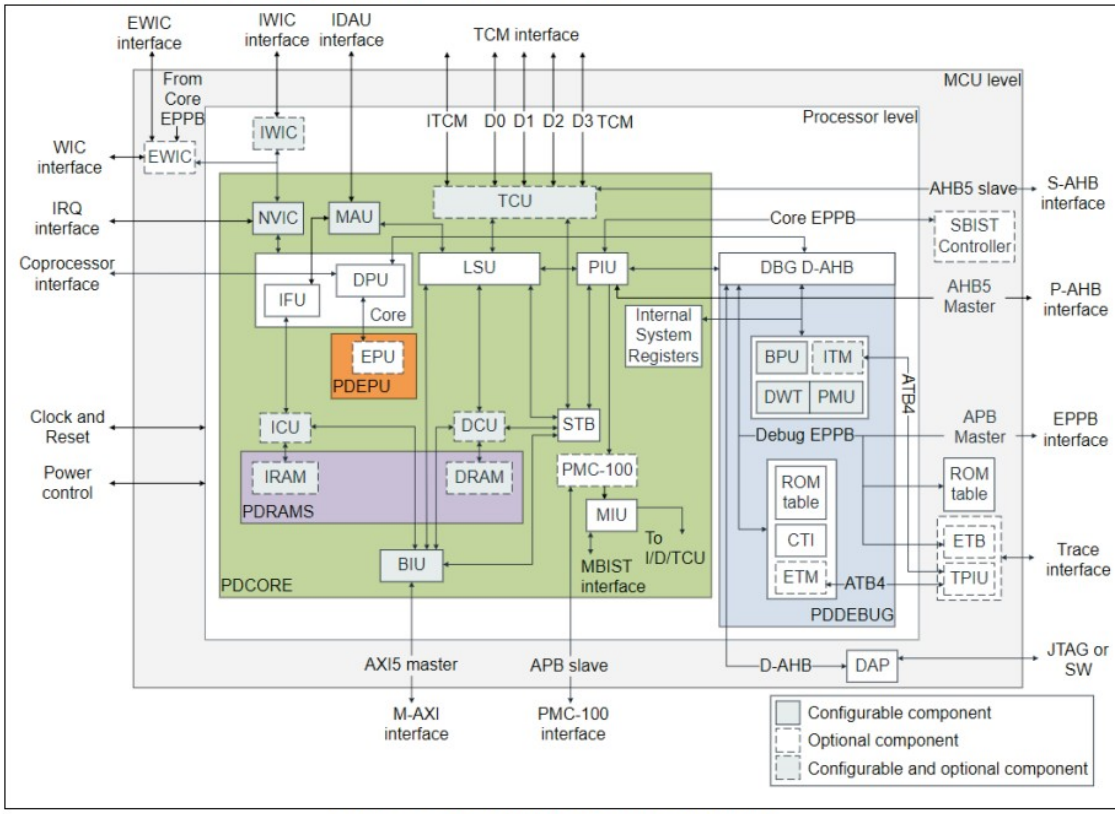


Figure 1. Cortex®-M85 Core Block Diagram

2.2 Renesas RA8 MCU

The RA8M1 MCU group incorporates a high-performance Arm® Cortex® -M85 core as shown in the previous section with Helium™ running up to 480 MHz with the following features.

- Up to 2 MB code flash memory
- 1 MB SRAM (128 KB of TCM RAM, 896 KB of user SRAM)
- Octal Serial Peripheral Interface (OSPI)
- Ethernet MAC Controller (ETHERC), USBFS, USBHS, SD/MMC Host Interface
- Analog peripherals
- Security and safety features.

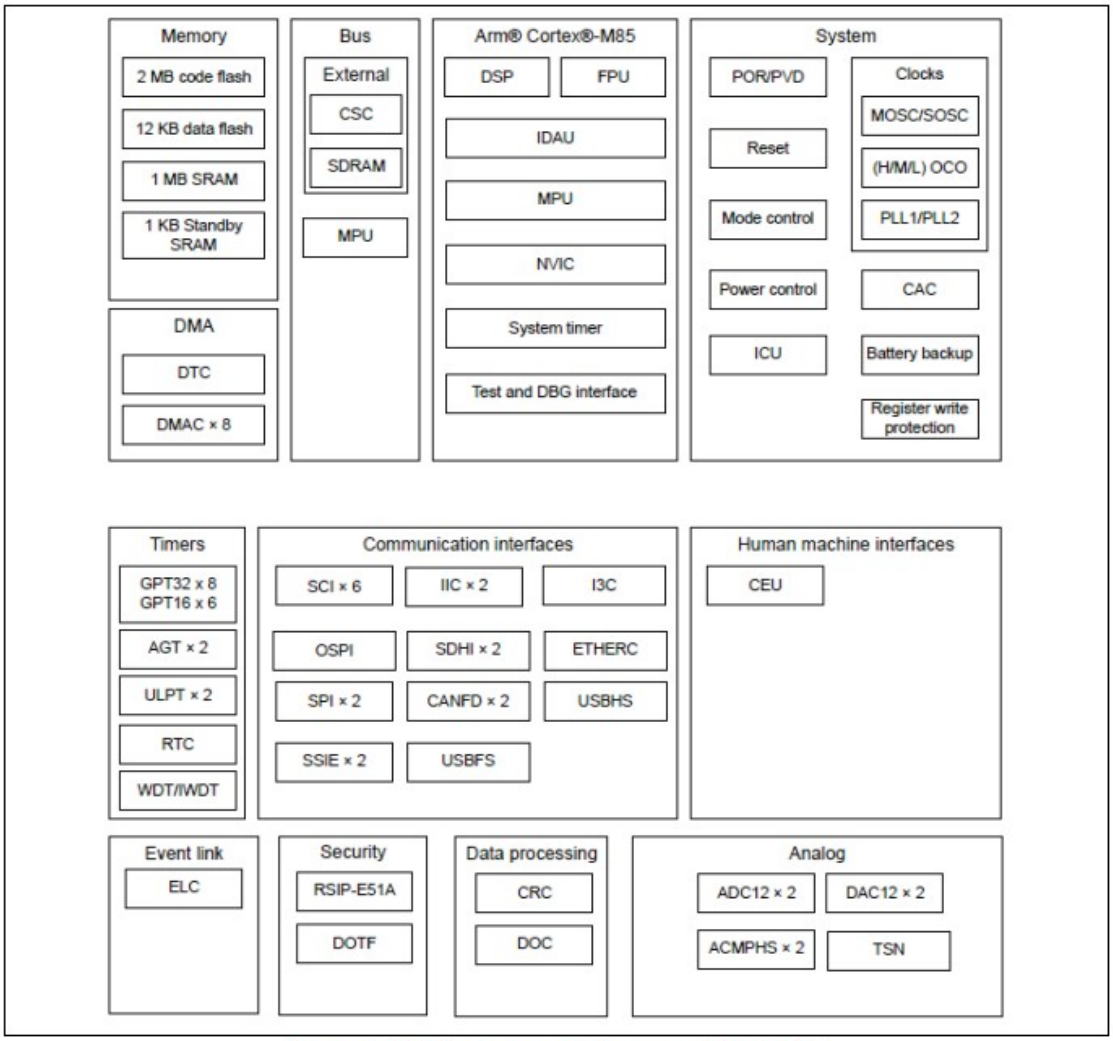


Figure 2. Block Diagram of Renesas RA8M1 MCU

2.3 Single Instruction Multiple Data

Most Arm® instructions are Single Instruction Single Data (SISD) instructions. The SISD instruction only operates on a single data item. It requires multiple instructions to process data items.

The Single Instruction Multiple Data (SIMD), on the other hand, performs the same operation on multiple items of same data type, concurrently. It means invoking/executing a single, multiple operations are being performed simultaneously.

Figure 3 shows the operation of VADD.I32 Qd, Qn, Qm instruction that adds the four pairs of 32-bit data together. Firstly, the four pairs of 32-bit input data are packed into separate lanes in two 128-bit Qn, Qm registers. Then, each lane in the 1st source register is then added to the corresponding lane in the 2nd source register. The results are stored in the same lane in the destination register Qd.

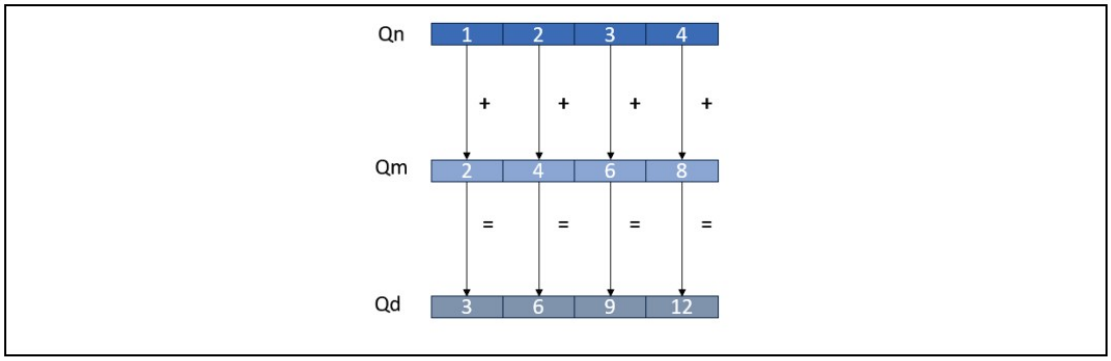


Figure 3. Operation of VADD.I32 Qd, Qn, Qm Instruction

2.4 Helium™ Applications

Digital Signal Processing (DSP) and Machine Learning (ML) are the main target applications for Helium™. Helium™ offers significant performance increases in these applications. Typically, Helium applications are created using Helium intrinsics.

Helium instructions are made available as intrinsic routines through the arm_mve.h in IAR EWARM installation, located in IAR Systems\Embedded Workbench x.x\arm\inc\c\aarch32. They give users access to the Helium instructions from C and C++ without the need to write assembly code.
Many functions in CMSIS-DSP and CMSIS-NN libraries have been optimized by Arm to use the Helium instructions instead. Renesas FSP supports both libraries, making it easier for users to develop applications based on these libraries. In the FSP configuration, select Arm DSP Library Source (CMSIS5-DSP version 5.9.0 or later) and Arm NN Library Source (CMSIS-NN version 4.1.0 or later) when generating projects to add CMSIS-DSP and CMSIS-NN supports to your project.

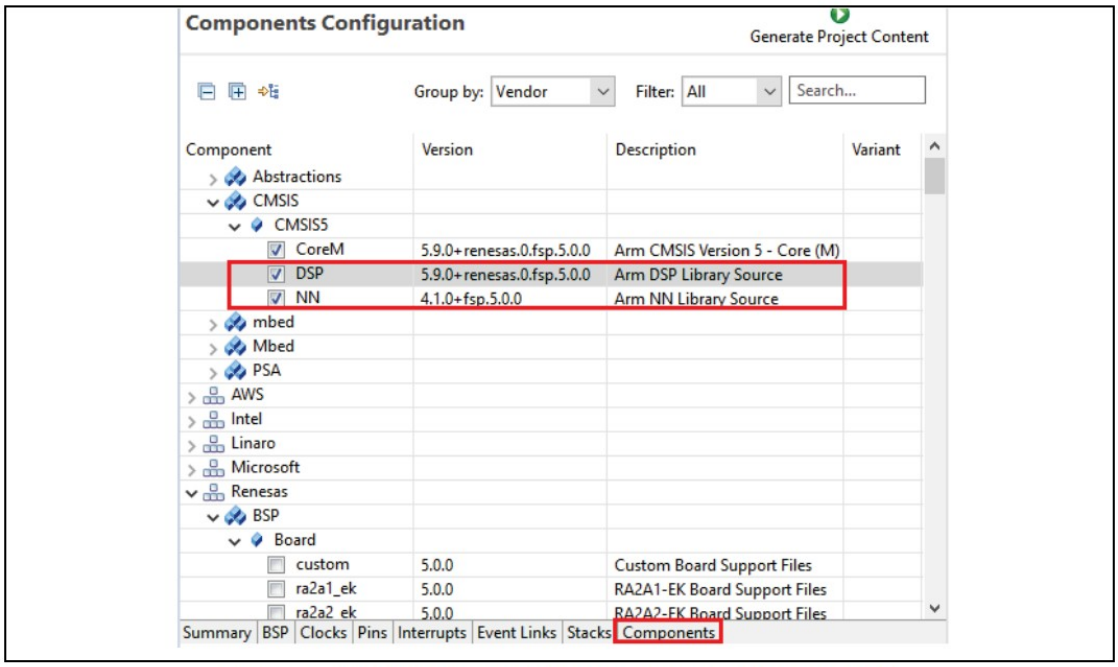


Figure 4. CMSIS-DSP and CMSIS-NN supports in Renesas FSP

CMSIS-DSP and CMSIS-NN can also be added using Stacks tab in FSP configurator, as shown below.

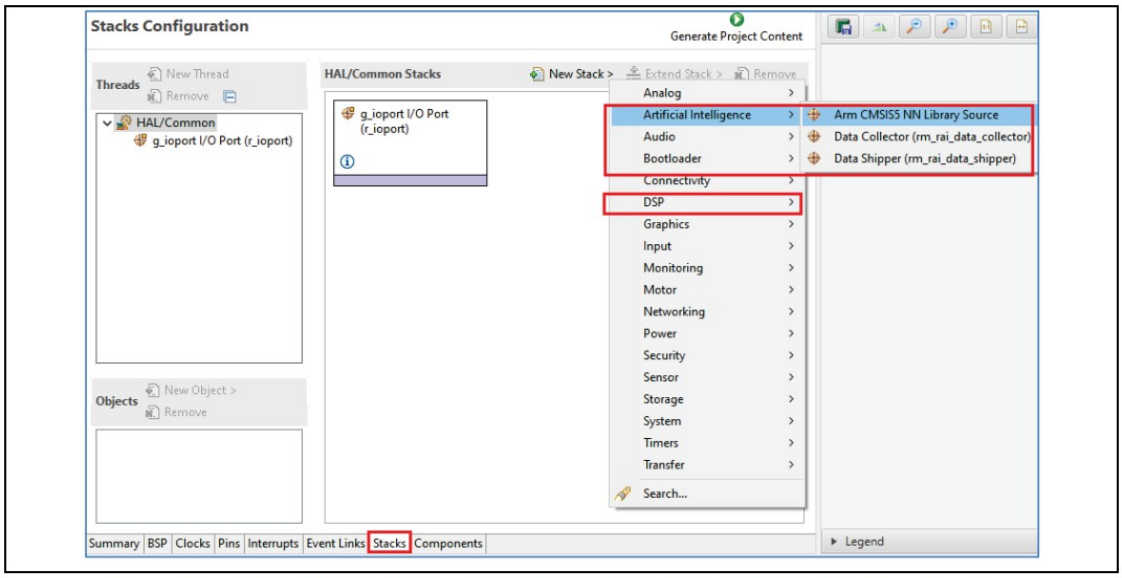


Figure 5. Adding CMSIS-DSP and CMSIS-NN Using Stacks Tab in FSP Configurator

Helium™ Support in Renesas FSP and IAR EWARM

IAR EWARM supports Helium™ instructions with the compiler settings. When generating a RA8M1 project using Renesas RA Smart Configurator and Flexible Software Package (FSP), CPU settings and software settings are pre-optimized for Cortex-M85 core and the CMSIS Helium™ support. Refer to the Renesas RA Smart Configurator Quick Start Guide for creating an IAR EWARM project for RA8 MCU.

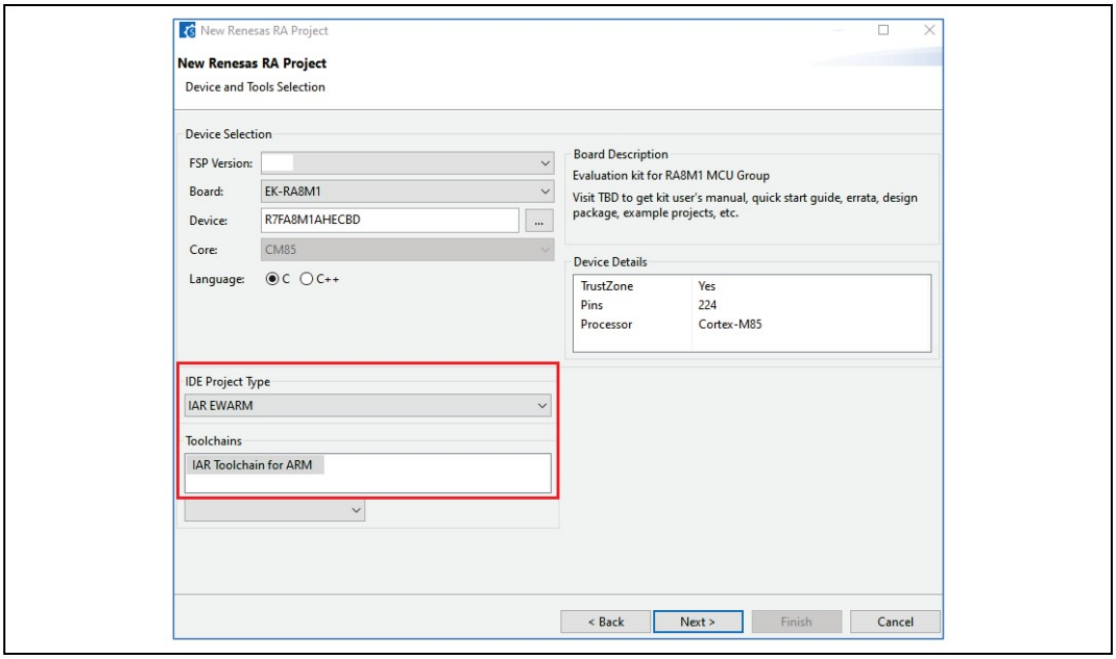


Figure 6. Create an EK-RA8M1 Project using Renesas RA Smart Configurator

The Cortex-M85 core will be selected in IAR EWARM settings, as shown below.

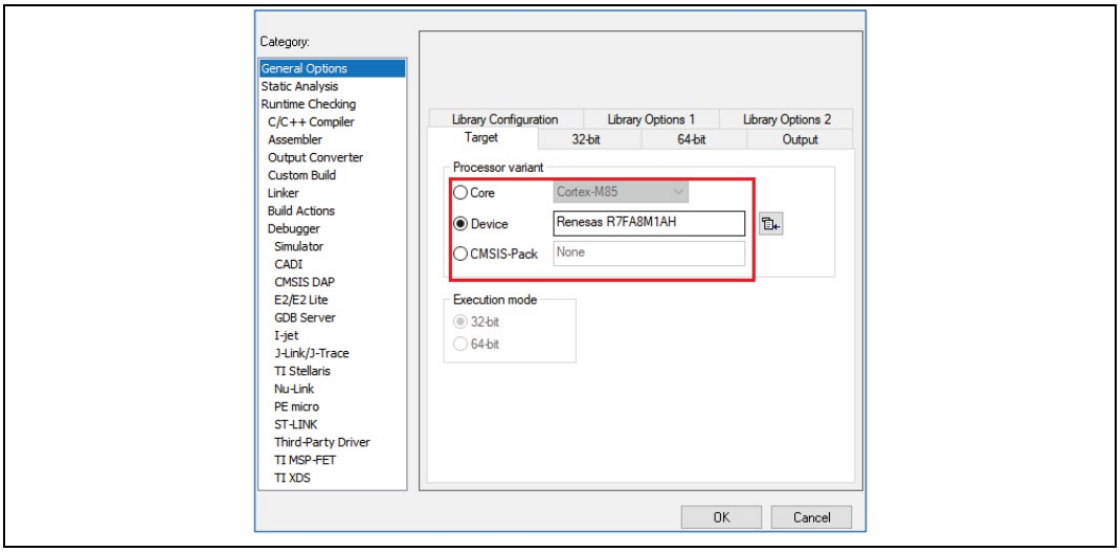


Figure 7. Confirm Project Settings on IAR EWARM

Check Project > Options > General Options to confirm if SIMD (NEON/HELIUM) is selected.

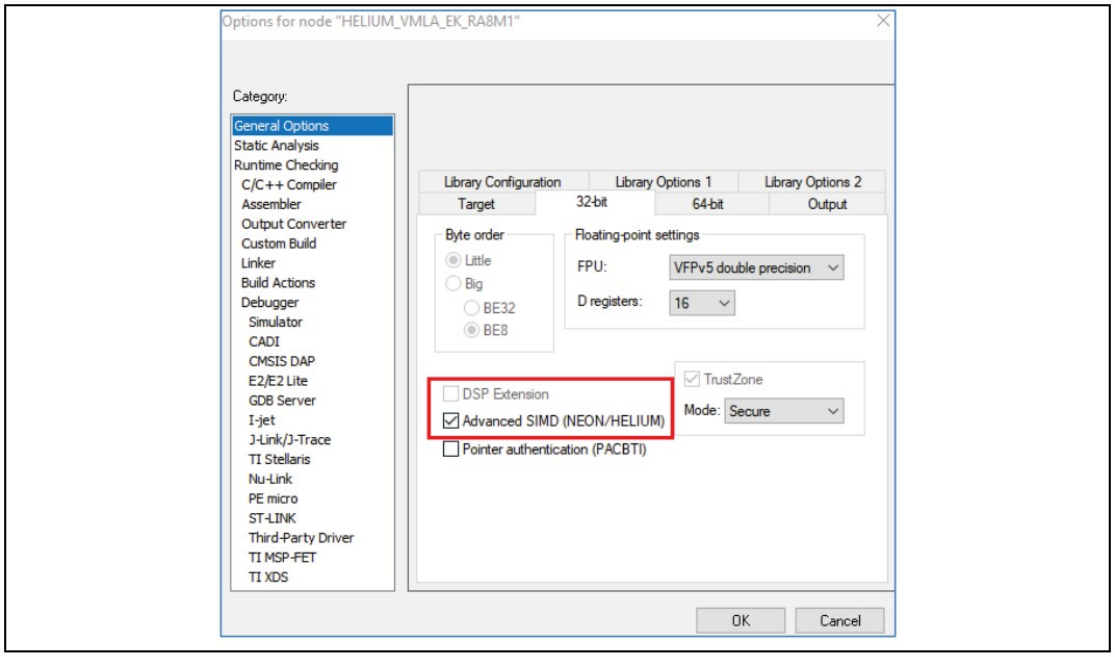


Figure 8. Example of Helium Selection in IAR EWARM

Even though, the project settings are pre-optimized for Cortex-M85, they can be customized if needed. Macro

definitions can be added to select project configurations to enable and disable some portions of the code in an IAR EWARM project. Go to Project > Options to change setups for the project if needed. The project settings can be confirmed using the Build Messages window on IAR EWARM. Some highlight settings for RA8 MCUs are marked in red below.

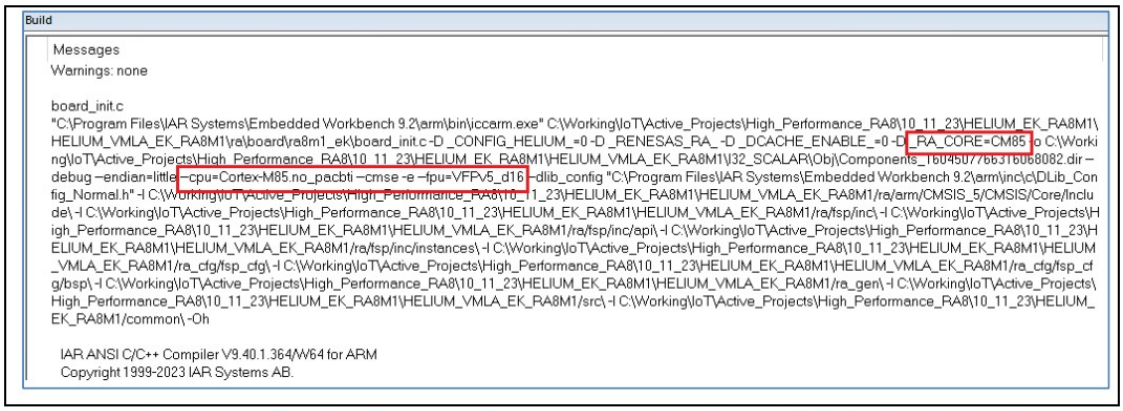


Figure 9. Example of Build Command on IAR EWARM

Application Project

There are three projects accompanying this application note. All have the scalar code equivalent to Helium functions.

- The Vector Multiply Accumulate (VMLA) and the scalar code equivalent.
- The Vector Multiply Accumulate Add Accumulate Across Vector (VMLADAVA) and the scalar code equivalent.
- The ARM DSP Dot Product function and the scalar code equivalent.

The projects are configured in various settings to utilize DTCM, ITCM, and cache to showcase the performance improvements of Helium technology compared to scalar code.

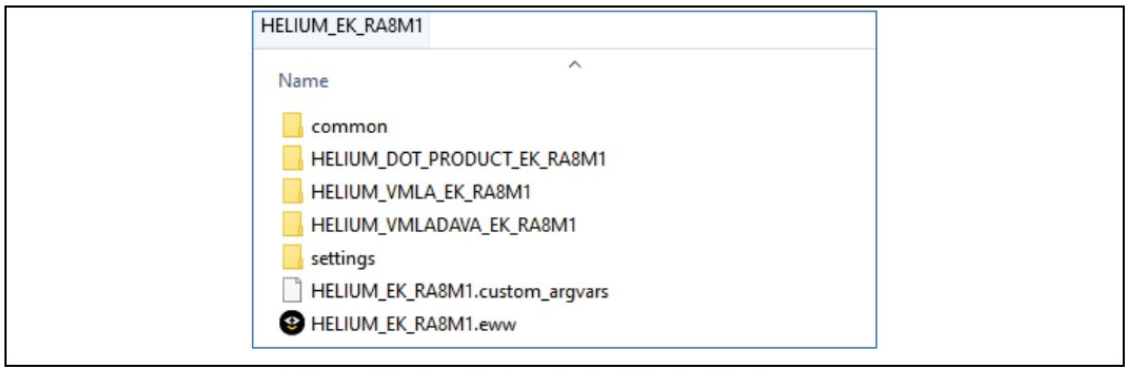


Figure 10. Application Projects in the Workspace

The available configuration for each project is as follows.

Project	HELIUM_VMLA_EK_RA8M1	HELIUM_VMLADAVA_EK_RA8M1	HELIUM_DOT_PRODUCT_EK_RA8M1
Configuration			
I32_SCALAR	✓	✓	✓
I32_HELIUM	✓	✓	✓
I32_HELIUM_DTCM	✓	✓	✓
I32_HELIUM_ITCM	✓	✓	

Figure 11. Configuration Available in Application Projects

Where I32_SCALAR is for the scalar code, I32_HELIUM is for the Helium code, I32_HELIUM_DTCM is for the Helium code that utilizes DTCM, and I32_HELIUM_ITCM is for the Helium code placed ITCM.
The projects in this application note are set to “High” and “Balanced” as shown in the following screenshot.

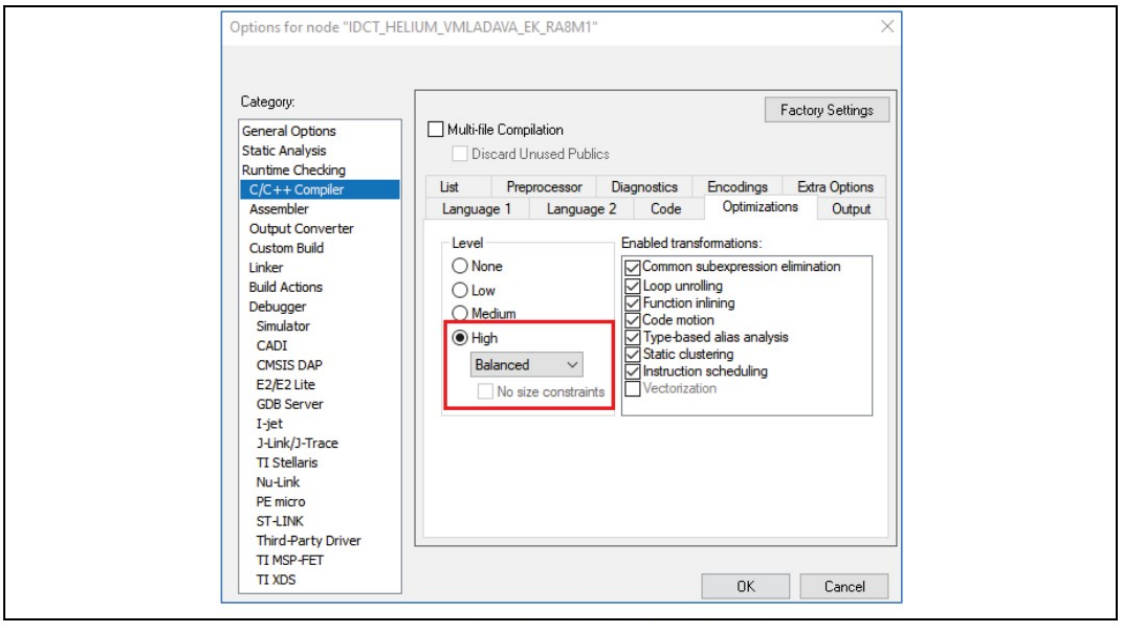


Figure 12. EWARM Compiler Optimization Setting

The _CONFIG_HELIUM_ symbol is preset to select scalar operation, Helium Operation, or enable the code to utilize DTCM and ITCM.

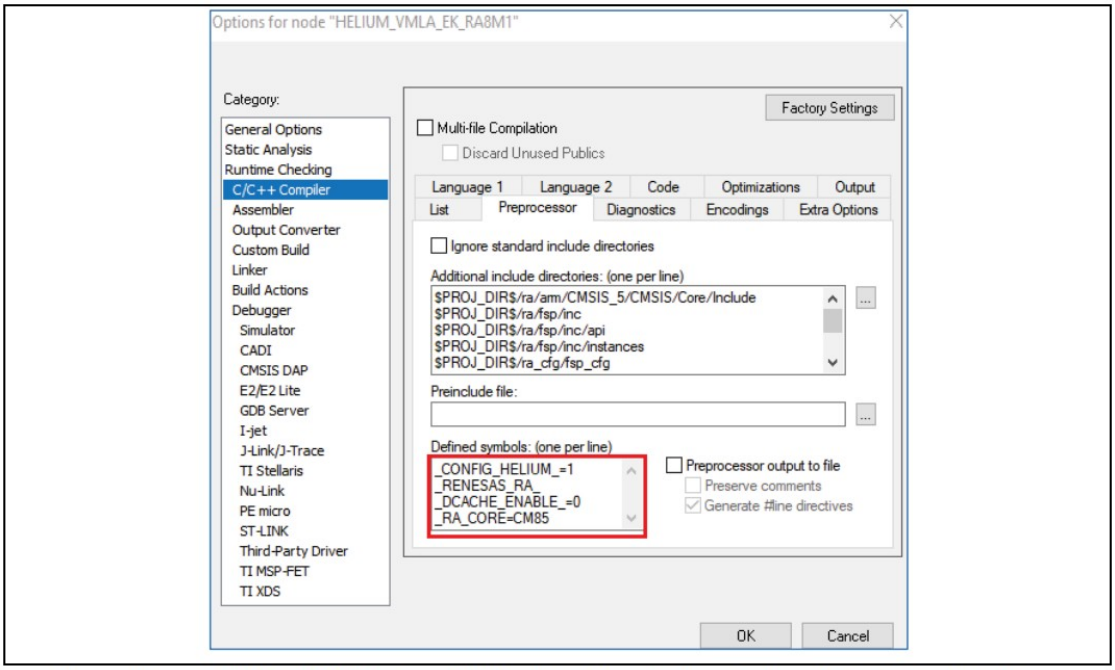


Figure 13. _CONFIG_HELIUM_ Symbol Used to Select Helium Code and Scalar Code Options

4.1 Vector Multiply Accumulate Instruction VMLA Example In VMLA instruction, each element in the input vector2 is multiplied by the scalar value. The result is added
to the respective element of input vector1. The results are stored in the destination register.
The steps of VMLA.S32 Qda, Qn, Rm instruction are shown in the following figure.

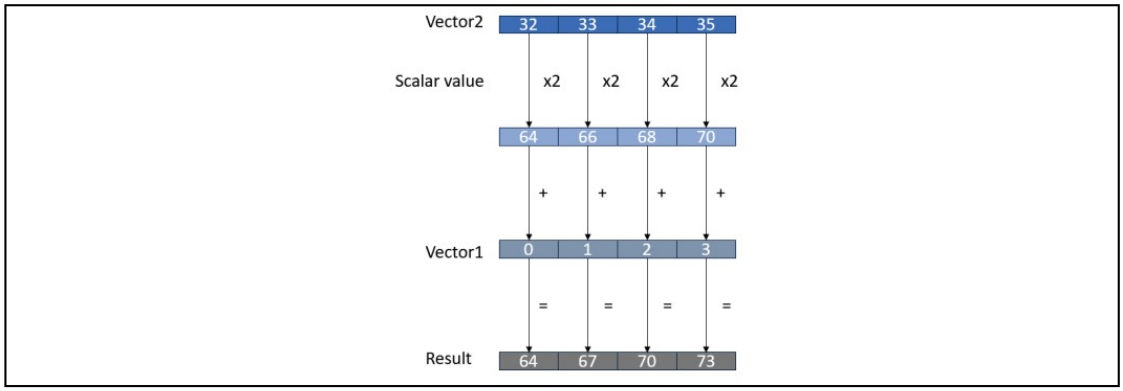


Figure 14. VMLA Operation

The intrinsic function vmlag_n_s32 in Figure 15 is used to showcase the performance of VMLA.S32 Qda, Qn, Rm instruction versus the scalar equivalent.

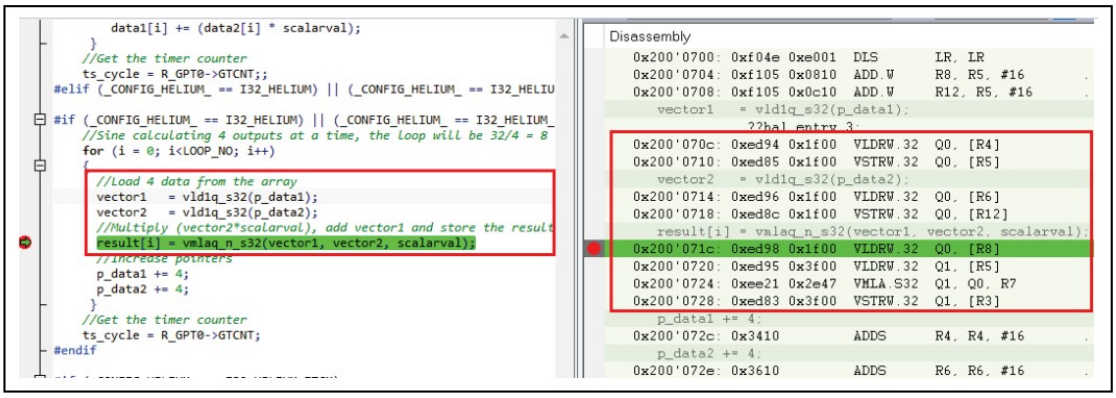


Figure 15. Example of VMLA Instruction Using Intrinsics and Disassembly Code

Figure 16 shows the scalar code equivalent to the Helium code in Figure 15.

Figure 16. Example of Scalar Code Equivalent of VMLA and Disassembly Code

4.2 Vector Instruction VMLADAVA Example

The VMLADAVA instruction multiplies the corresponding lanes of two input vectors, then sums these individual results to a produce a single value.

The steps of VMLADAVA.S32 Rda, Qn, Qm instruction are shown in the following figure.

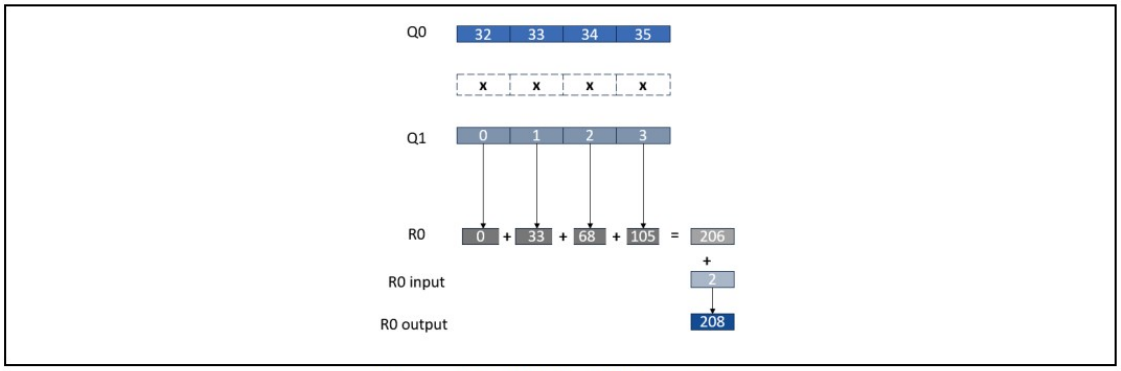


Figure 17. VMLADAVA Operation

The intrinsic function vmladavaq_s32 in Figure 18 is used to showcase the performance of VMLADAVA.S32 Rda, Qn, Qm instruction versus the scalar equivalent.

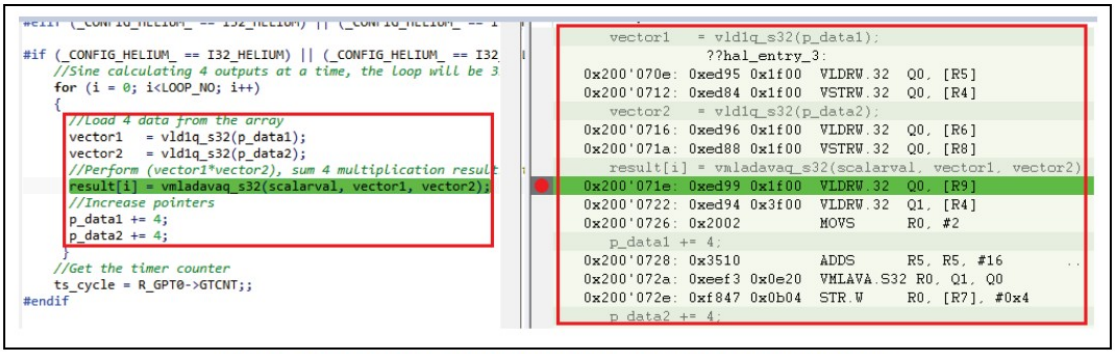


Figure 18. Example of VMLADAVA Instruction Using Intrinsics

Figure 19 shows the scalar code equivalent to the Helium™ code in Figure 18.

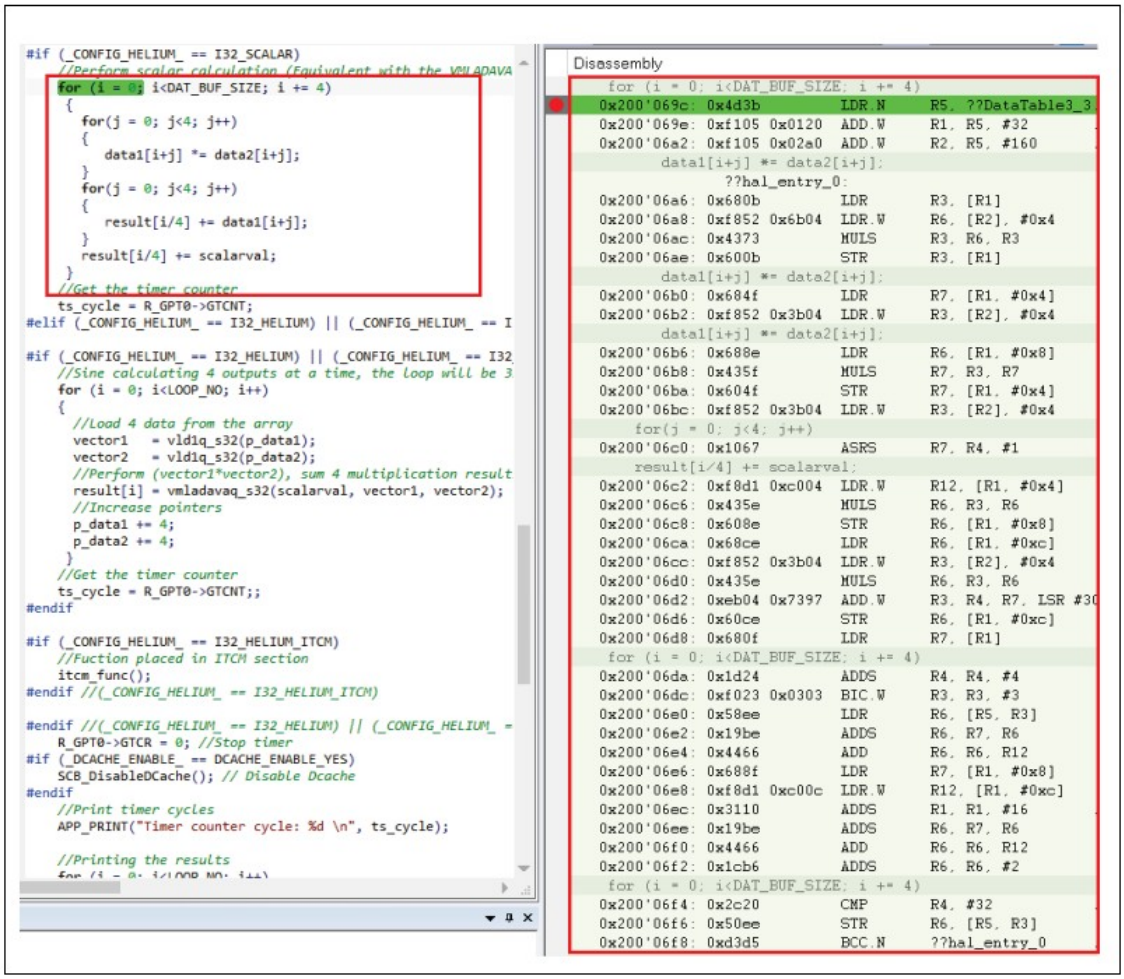


Figure 19. Example of Scalar Code Equivalent of VMLADAVA Instruction and Disassembly Code

4.3 ARM DSP Dot Product Example

The dot product example uses the arm_dot_product_f32 function in the Arm DSP library to calculate the dot product of two input vectors by multiplying element by element and sum them up. The performance of the Helium version of arm_dot_product_f32 will be compared with its scalar version.

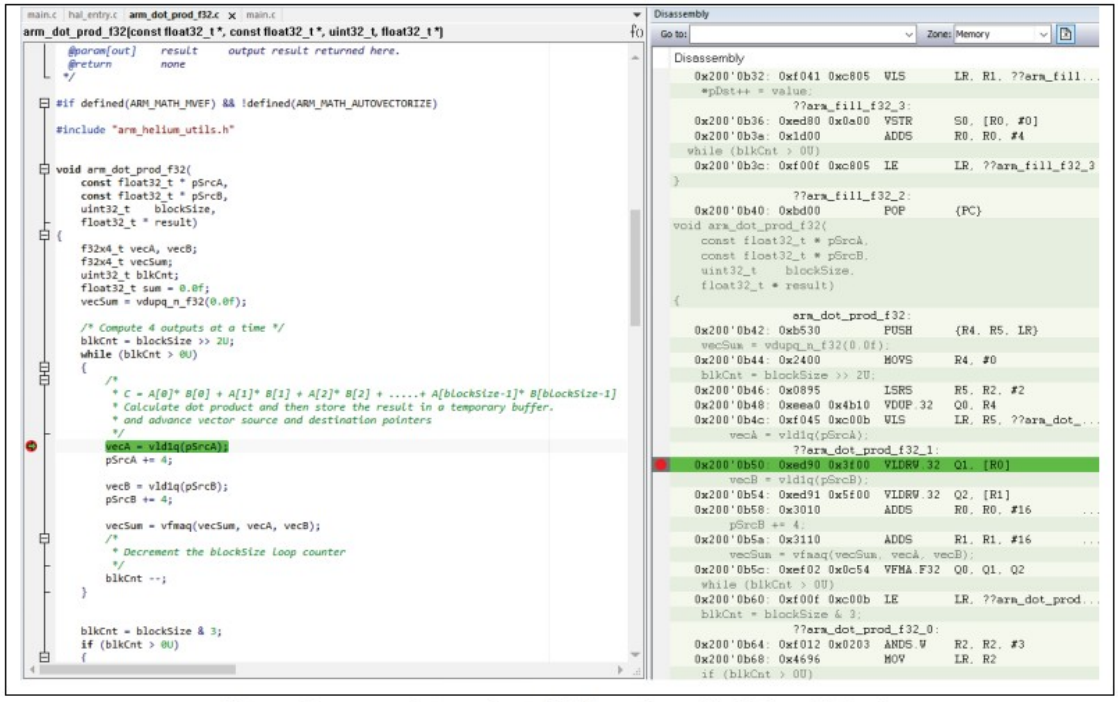


Figure 20. arm_dot_product_f32 Function with Helium™ Code

Renesas Flexible Software Package FSP supports Arm DSP Library Source for Cortex-M85 that uses Helium intrinsics. It will improve performance significantly compared to scalar code. Select Arm DSP Library Source in Project Configurator to add the DSP source to your project, as shown in Figure 21.

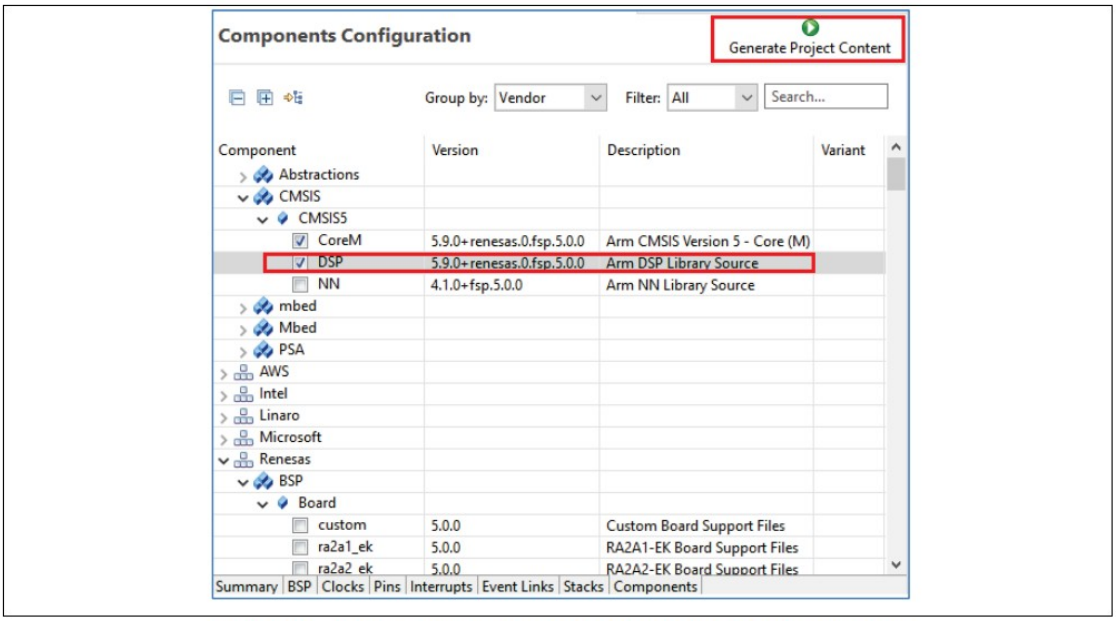


Figure 21. Adding Arm Library DSP Source in FSP Configurator

Click Generate Project Content, the Arm DSP library source will be added to the project.

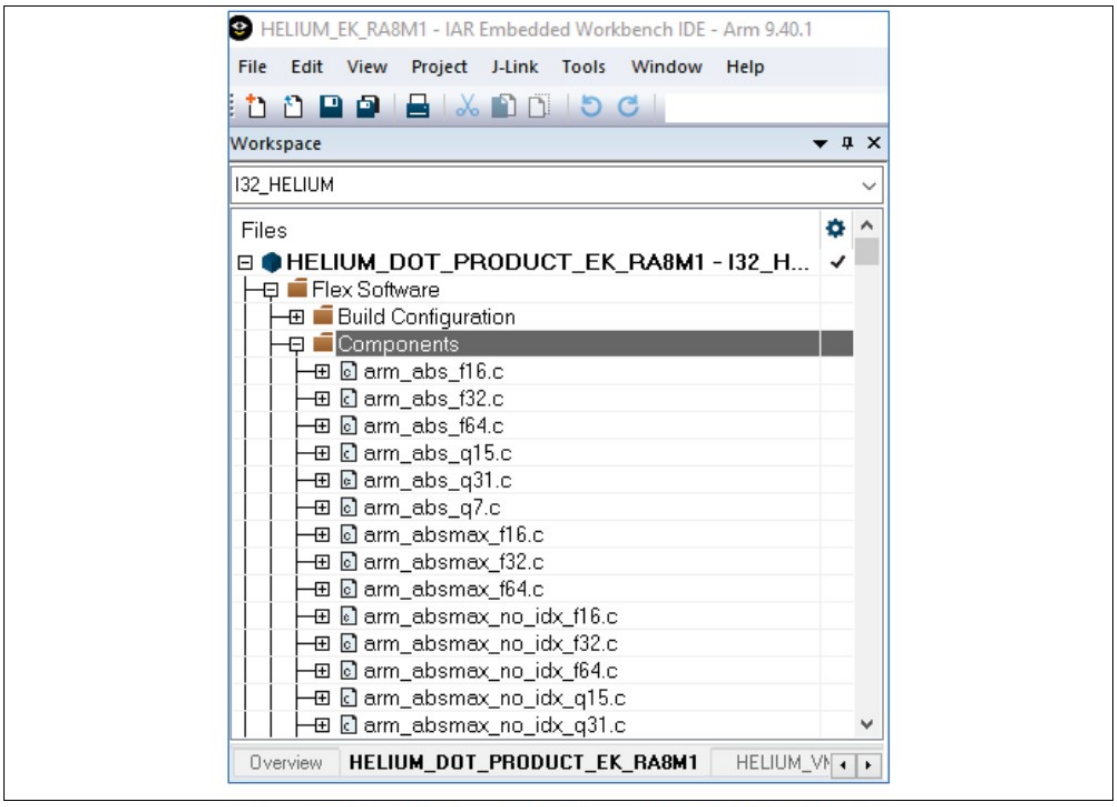


Figure 22. Arm Library DSP Source Added in FSP Project

4.4 Performance Improvement

You can utilize Tightly Coupled Memory (TCM) and Cache together with Helium™ to achieve higher performance. Typically, TCM provides single-cycle access and avoids delays in data access. Critical routines and data can be placed in TCM areas to ensure faster access. TCM does not use caches.

4.4.1 Tightly Coupled Memory (TCM)

The 128 KB TCM memory in RA8 MCU consists of 64 KB ITCM (Instruction TCM) and 64 KB DTCM (Data TCM). Note that accessing TCM is not available in CPU Deep Sleep mode, Software Standby mode, and Deep Software Standby mode.

Figure 23 shows ITCM and DTCM in the Local CPU Subsystem.

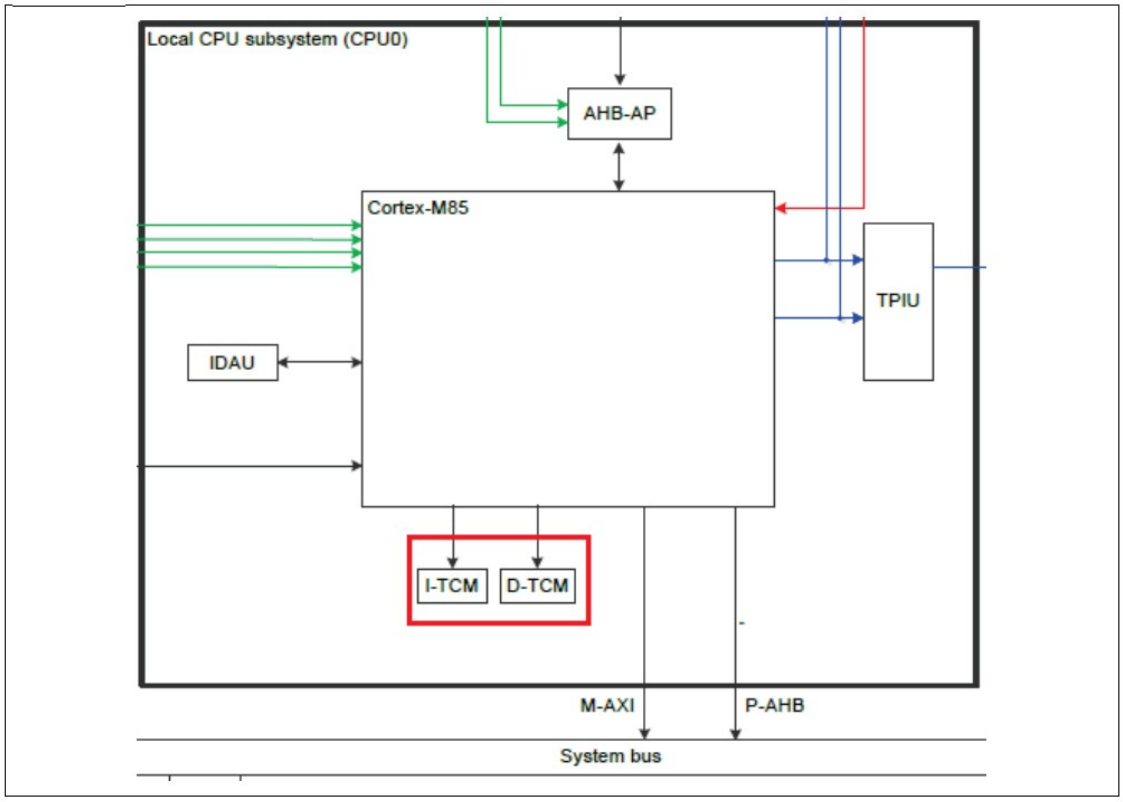


Figure 23. ITCM and DTCM in Local CPU Subsystem

FSP initializes both ITCM and DTCM areas by default. The linker script has defined sections for ITCM and DTCM areas, making it easy to utilize in user applications.

Figure 24 and Figure 25 are snapshots of ITCM and DCTM locations in RA8 MCU.

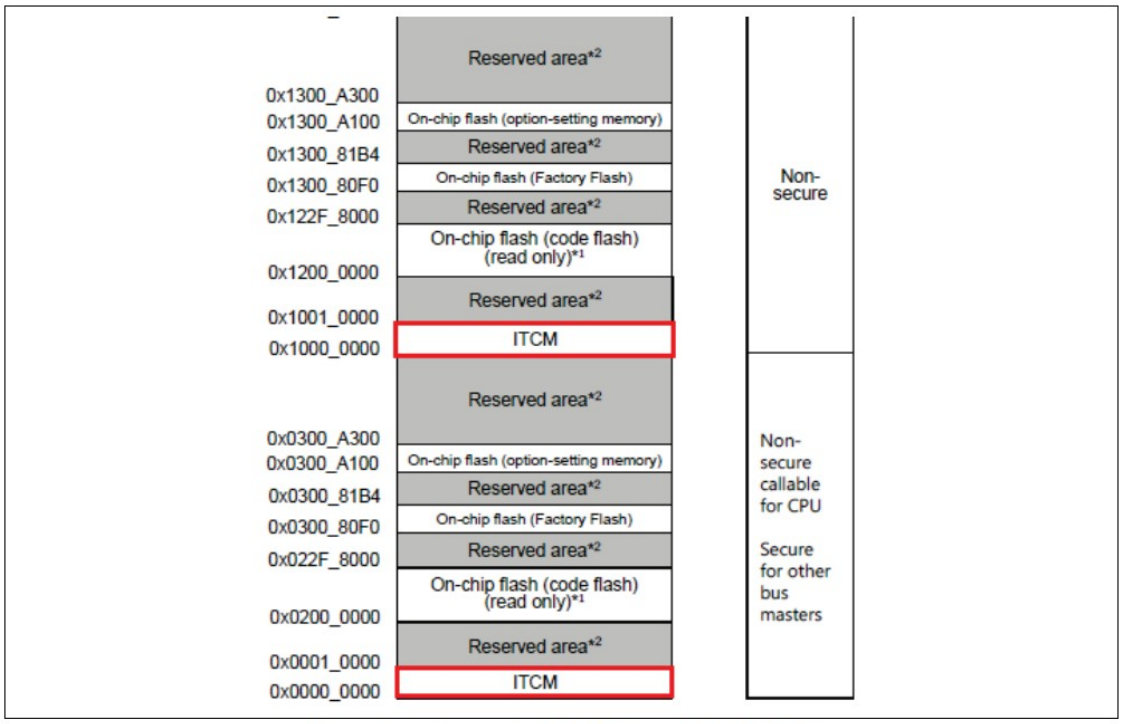


Figure 24. Example of ITCM Areas in RA8 MCU

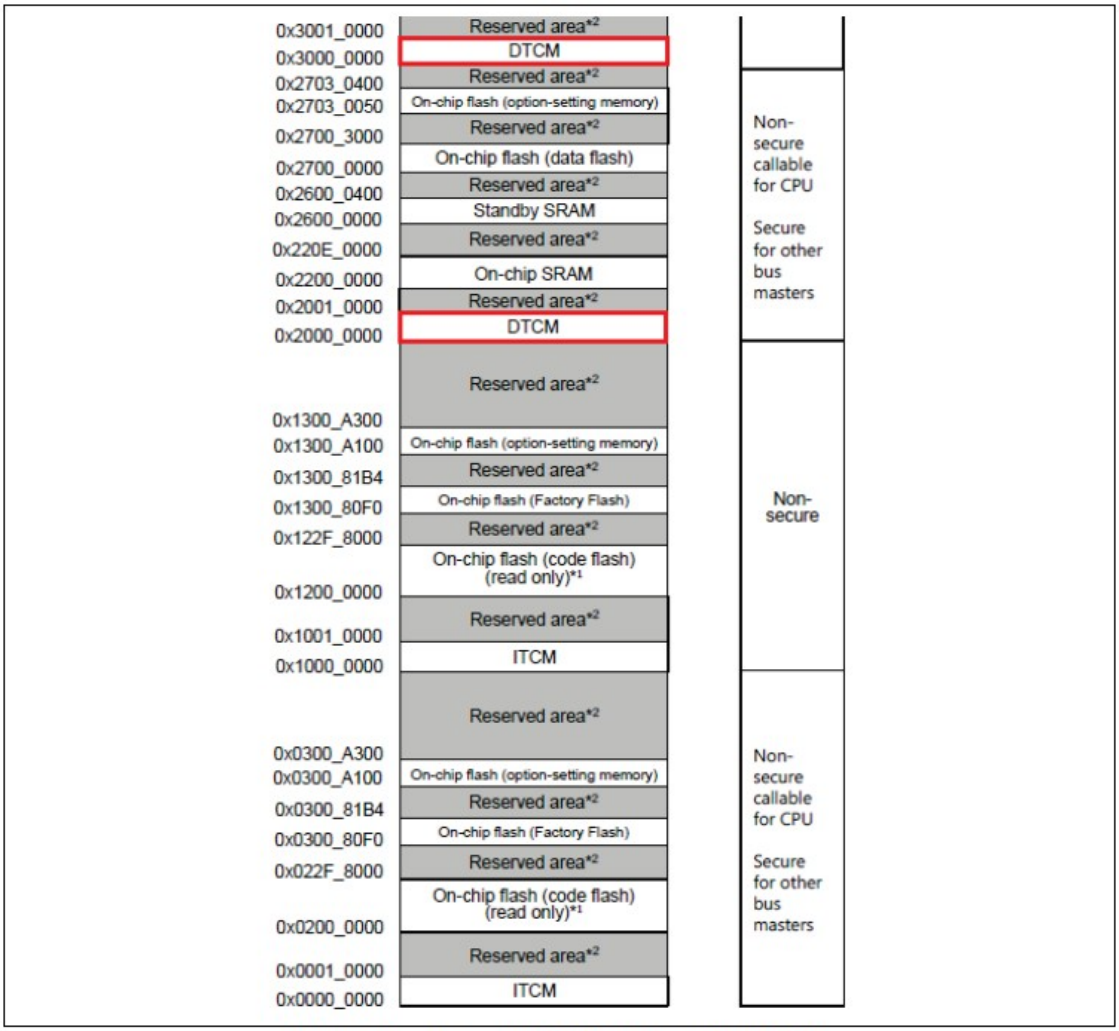


Figure 25. Example of DTCM Areas in RA8 MCU

4.4.2 Improve Performance Using DTCM

You can place data in the DTCM section (.dtcm_data) in an FSP-based project using the _attribute_ directive, as shown in Figure 26.

```
#if (_CONFIG_HELIUM_ == I32_HELIUM) || (_CONFIG_HELIUM_ == I32_HELIUM_ITCM)
static int32x4_t vector1;
static int32x4_t vector2;
static int32x4_t result[8];
#elif ( CONFIG_HELIUM  == I32_HELIUM_DTCM)
static int32x4_t __attribute__((section(".dtcm_data"))) vector1;
static int32x4_t __attribute__((section(".dtcm_data"))) vector2;
static int32x4_t __attribute__((section(".dtcm_data"))) result[8];
#endif
```

Figure 26. Placing Variables in DTCM Section

The above data placement can be confirmed using the memory map generated by the compiler.

```
g_bsp_rom_pbps_sec3      0x300'a26c    0x4  Data  Lc  bsp_rom_registers.o [1]
g_bsp_rom_sas            0x300'a134    0x4  Data  Lc  bsp_rom_registers.o [1]
g_clock_freq            0x2200'0e10   0x2c  Data  Lc  bsp_clocks.o [1]
g_fsp_version            0x200'11dc    0x4  Data  Lc  bsp_common.o [1]
g_fsp_version_build_string
                         0x200'11e8   0x44  Data  Lc  bsp_common.o [1]
g_fsp_version_string     0x200'11e0    0x8  Data  Lc  bsp_common.o [1]
g_interrupt_event_link_select
                         0x200'0040   0xc0  Data  Wk  bsp_irq.o [1]
g_ioport_ctrl           0x2200'014c    0x8  Data  Gb  common_data.o [2]
g_main_stack            0x2200'0a10  0x400  Data  Lc  startup.o [1]
g_prcr_masks             0x200'0604    0x8  Data  Lc  bsp_register_protection.o [1]
g_protect_counters      0x2200'0144    0x8  Data  Gb  bsp_register_protection.o [1]
g_protect_pfswe_counter
                        0x2200'0140    0x4  Data  Gb  bsp_io.o [1]
g_vbatt_pins_input       0x200'0d64    0x8  Data  Lc  r_ioport.o [1]
hal_entry                0x200'067d   0xe2  Code  Gb  hal_entry.o [3]
main                     0x200'1191    0xa  Code  Gb  main.o [2]
r_ioport_pfs_write       0x200'0c5b   0x2e  Code  Lc  r_ioport.o [1]
r_ioport_pins_config     0x200'0c2b   0x30  Code  Lc  r_ioport.o [1]
result                  0x2000'0020   0x80  Data  Lc  hal_entry.o [3]
vector1                 0x2000'0000   0x10  Data  Lc  hal_entry.o [3]
vector2                 0x2000'0010   0x10  Data  Lc  hal_entry.o [3]
```

Figure 27. Example of Variables Placed in DTCM Area in Memory Map

4.4.3 Improve Performance Using ITCM

One of the methods to place some portions of the code in the ITCM section (.itcm_data) is using the #Pragma directive, as shown in Figure 28.

```
#if (_CONFIG_HELIUM_ == I32_HELIUM_ITCM)
//Placing functions in section .itcm_data
#pragma default_function_attributes = @ ".itcm_data"
void itcm_func(void)
{
    int i;
    //Pointer values for both arrays
    int32_t *p_data1  = &data1[0];
    int32_t *p_data2  = &data2[0];
    R_GPT0->GTCR  = 0; // Stop timer
    R_GPT0->GTCNT = 0; // Clear counter
    R_GPT0->GTCR  = 1; // Start timer
    //Sine calculating 4 outputs at a time, the loop will be 32/4 = 8 (LOOP_NO )
    for (i = 0; i<LOOP_NO; i++)
    {
      //Load 4 data from the array
      vector1   = vld1q_s32(p_data1);
      vector2   = vld1q_s32(p_data2);
      //Multiply (vector2*scalarval), add vector1 and store the results
      result[i] = vmlaq_n_s32(vector1, vector2, scalarval);
      //Increase pointers
      p_data1 += 4;
      p_data2 += 4;
     }
     //Get the timer counter
     ts_cycle = R_GPT0->GTCNT;
}
//End placing functions in section .itcm_data
#pragma default_function_attributes =
#endif
```

Figure 28. Example of Placing a Function in ITCM Section in IAR EWARM Project

You can confirm code placement using the .map file generated by the compiler or using the Disassembly Window on the debugger.

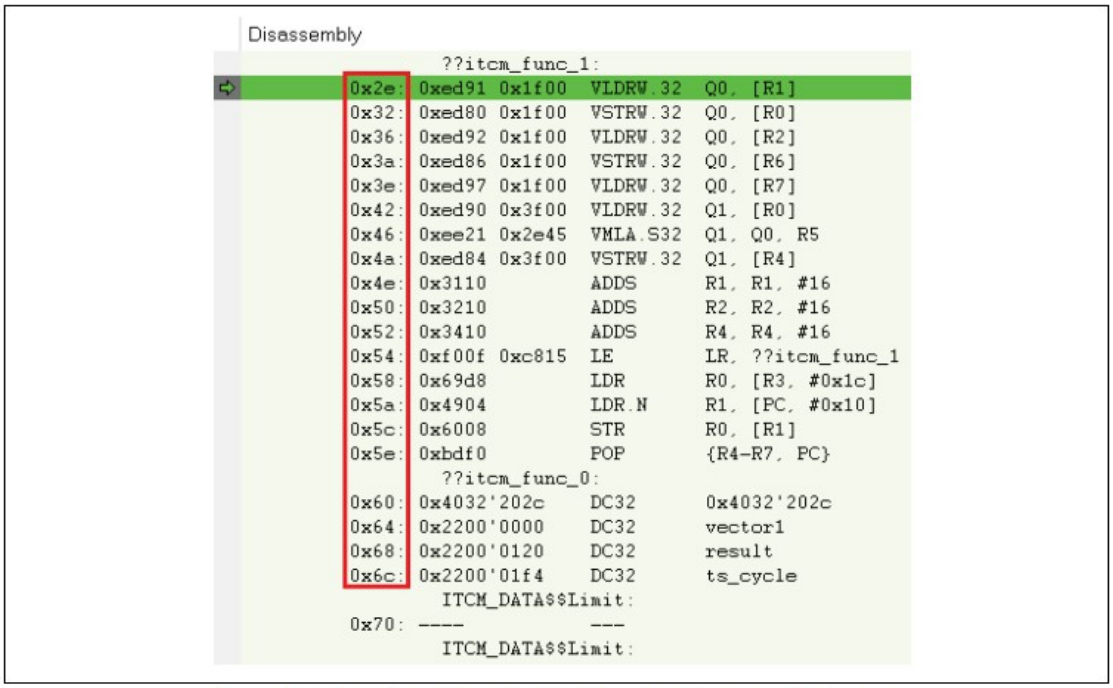


Figure 29. Function Placed in ITCM Section Shown on Debugger

4.5 Improve Performance by Utilizing Data Cache

When a function utilizes long loops, it executes the same code repeatedly. Furthermore, in many applications, data access may be repeated and sequential. Performance in these scenarios can improve significantly with cache enabled.

In FSP, the instruction cache enable is done in a function named SystemInit in system.c, as shown in Figure 30 and Figure 31.

```
/***********************************************************************************************************************
 * Macro definitions
 **********************************************************************************************************************/

/* Mask to select CP bits( 0xF00000 ) */
#define CP_MASK                                 (0xFU << 20)

/* Startup value for CCR to enable instruction cache, branch prediction and LOB extension */
#define CCR_CACHE_ENABLE                        (0x000E0201)

/* Value to write to OAD register of MPU stack monitor to enable NMI when a stack overflow is detected. */
#define BSP_STACK_POINTER_MONITOR_NMI_ON_DETECTION    (0xA500U)
```

Figure 30. Macro Definition to Enable Cache in system.c in FSP

```
/***********************************************************************************************************************
 * Initialize the MCU and the runtime environment.
 **********************************************************************************************************************/
void SystemInit (void)
{
#if defined(RENESAS_CORTEX_M85)

    /* Enable the ARM core instruction cache, branch prediction and low-overhead-branch extension.
     * See Section 5.5 of the Cortex-M55 TRM and Section D1.2.9 in the ARMv8-M Architecture Reference Manual */
    SCB->CCR = (uint32_t) CCR_CACHE_ENABLE;
    __DSB();
    __ISB();
#endif

    /* Enable the ARM core instruction cache, branch prediction and low-overhead-branch extension.
     * See Section 5.5 of the Cortex-M55 TRM and Section D1.2.9 in the ARMv8-M Architecture Reference Manual */
    SCB->CCR = (uint32_t) CCR_CACHE_ENABLE;
```

Figure 31. Code to Enable Instruction Cache in FSP

Figure 31. Code to Enable Instruction Cache in FSP

The application projects have a setting to enable data cache. Set the _DCACHE_ENABLE_ symbol in the project option to 1 to enable data cache. Even though data cache improves performance, it can cause concurrency and coherency issues. It is good practice to enable the cache for application code that has repeated access to the same set of data.

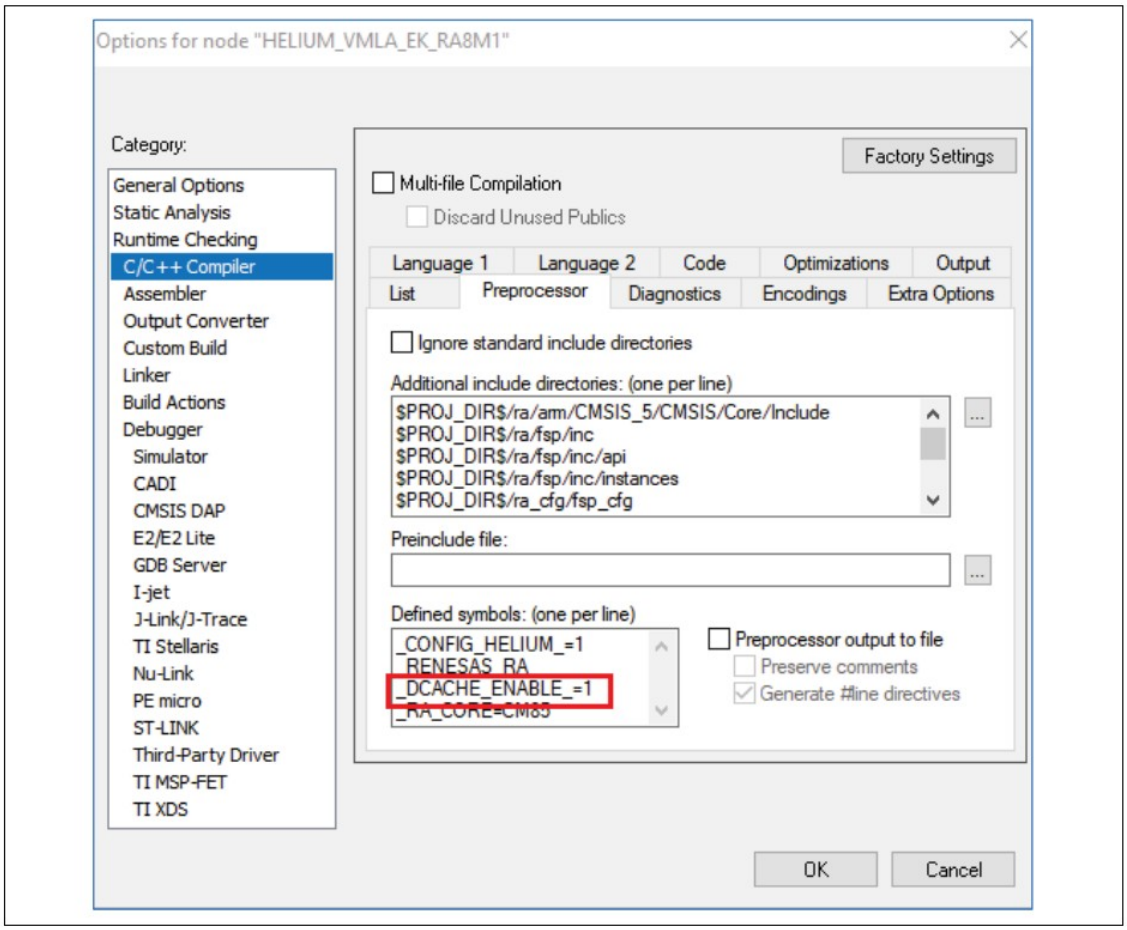


Figure 32. Example of Data Cache Enable in Application Project

Example code to enable and disable data cache are shown in Figure 33 and Figure 34.

```
#if (_DCACHE_ENABLE_ == DCACHE_ENABLE_YES)
    SCB_EnableDCache(); //Enable DCache
#endif
```

Figure 33. Example Code to Enable DCACHE

```
#if (_DCACHE_ENABLE_ == DCACHE_ENABLE_YES)
    SCB_DisableDCache(); // Disable Dcache
#endif
```

Figure 34. Example Code to Disable DCACHE

Another method to enable data cache is using FSP Configurator: BSP > Properties > Settings > MCU (RA8M1) Family > Cache settings > Data cache, as shown in Figure 35.

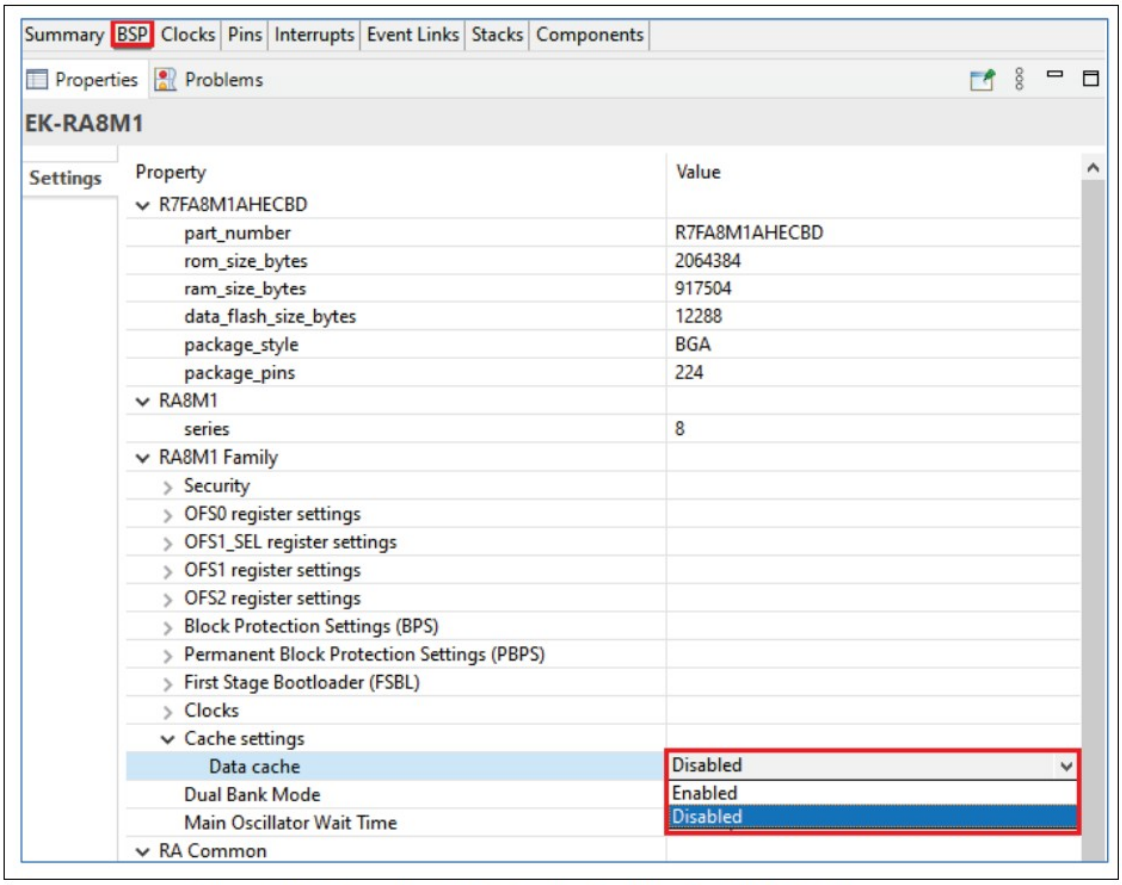


Figure 35. Example of Data Cache Enable using FSP Configurator

4.6 Using General Purpose (GPT) Timer for Benchmarking

In the projects, GPT0 timer is used to measure time for performance benchmarking.

```
    //Clear and start timer for benchmarking
    R_BSP_MODULE_START(FSP_IP_GPT, 0);
    R_GPT0->GTCNT = 0; // Clear counter
    R_GPT0->GTCR  = 1; // Start timer

#if (_CONFIG_HELIUM_ == I32_SCALAR)
    //Perform scalar calculation (Equivalent with the multiply accumulate instruction)
    for (i = 0; i<DAT_BUF_SIZE; i++)
     {
        data1[i] += (data2[i] * scalarval);
     }
    //Get the timer counter
    ts_cycle = R_GPT0->GTCNT;;
#elif (_CONFIG_HELIUM_ == I32_HELIUM) || (_CONFIG_HELIUM_ == I32_HELIUM_DTCM) || (_CONFIG_HELIUM_ == I32_HELIUM_ITCM)

#if (_CONFIG_HELIUM_ == I32_HELIUM) || (_CONFIG_HELIUM_ == I32_HELIUM_DTCM)
    //Sine calculating 4 outputs at a time, the loop will be 32/4 = 8 (LOOP_NO )
    for (i = 0; i<LOOP_NO; i++)
    {
      //Load 4 data from the array
      vector1   = vld1q_s32(p_data1);
      vector2   = vld1q_s32(p_data2);
      //Multiply (vector2*scalarval), add vector1 and store the results
      result[i] = vmlaq_n_s32(vector1, vector2, scalarval);
      //Increase pointers
      p_data1 += 4;
      p_data2 += 4;
     }
    //Get the timer counter
    ts_cycle = R_GPT0->GTCNT;
```

Figure 36. Example of the Timer Code for Benchmarking

Verify the Project

5.1 Open Project Workspace

The software tools required to run the application projects are as follows:

- IAR Embedded Workbench (IAR EWARM) version 9.40.1.63915 or later
- Renesas Flexible Software Package (FSP) v5.0.0 or later
- SEGGER RTT Viewer v7.92j or later

From IAR EWARM, open the HELIUM_EK_RA8M1.eww.

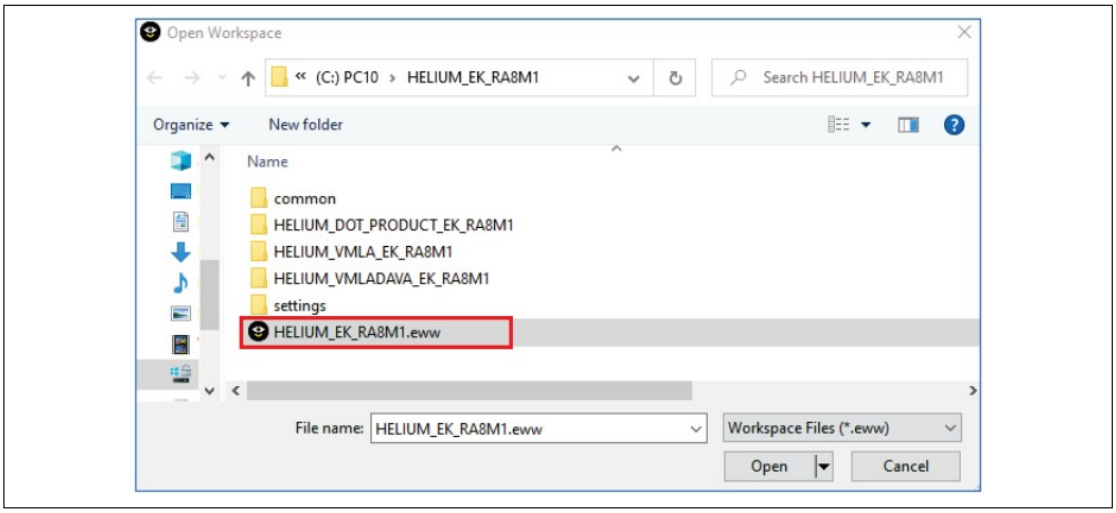


Figure 37. HELIUM_EK_RA8M1.eww Workspace

The HELIUM_EK_RA8M1 workspace consists of three projects named HELIUM_VMLA_EK_RA8M1, HELIUM_VMLADAVA_EK_RA8M1 and HELIUM_DOT_PRODUCT_EK_RA8M1.
Three projects that appear on the workspace when it opens, as shown in Figure 38.

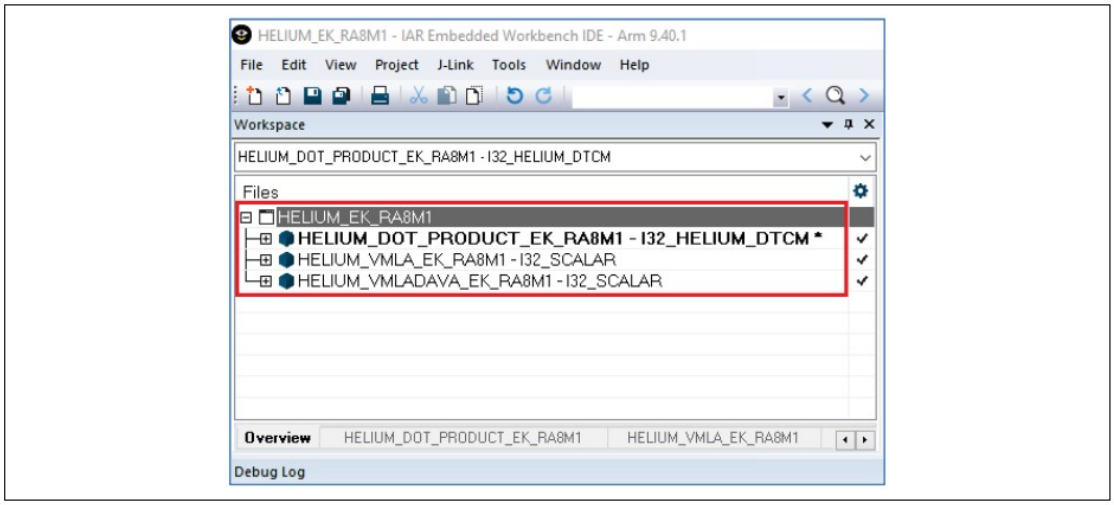


Figure 38. Projects are Opened in IAR EWARM

To enable data cache support in the application project, change _DCACHE_ENABLE_ symbols in Options > Preprocessor from 0 to 1, as shown in Figure 39.

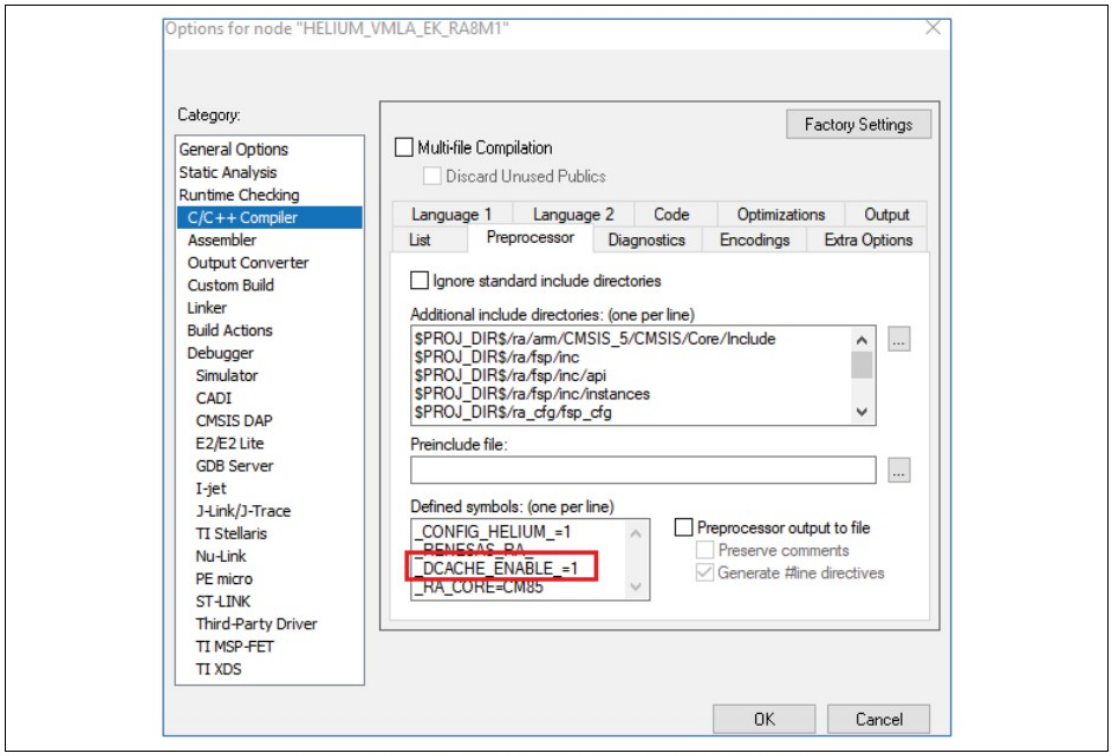


Figure 39. Enable Data Cache Support in Project

5.2 Build Project

There are several configurations in each project. Select a project, then a project configuration you wish to run before going to the next step.

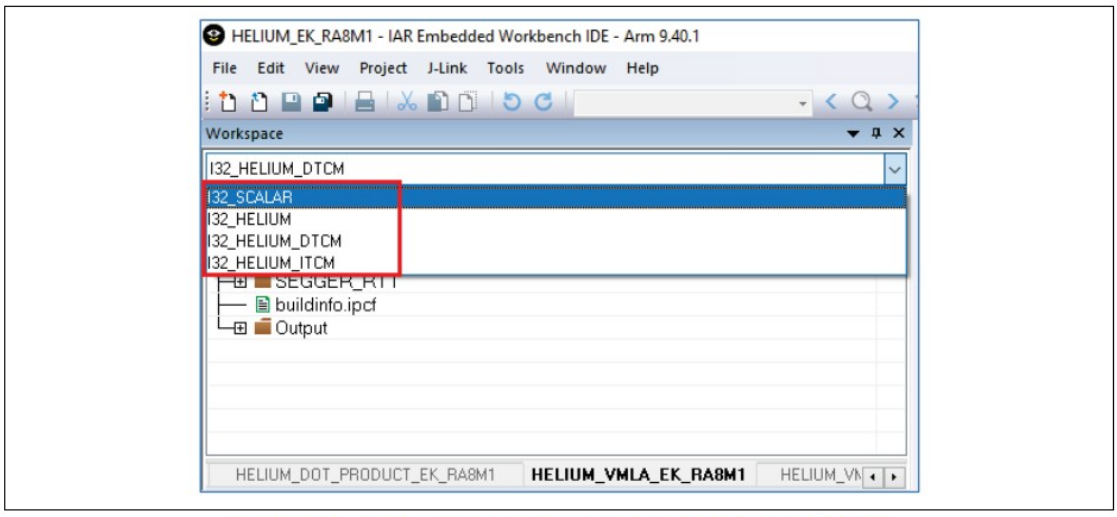


Figure 40. Cortex-M85 Configuration Control Register (CCR)

On IAR EWARM, launch RA Smart Configurator from Tools > RA Smart Configurator, and click “Generate Project Content” to generate project content.

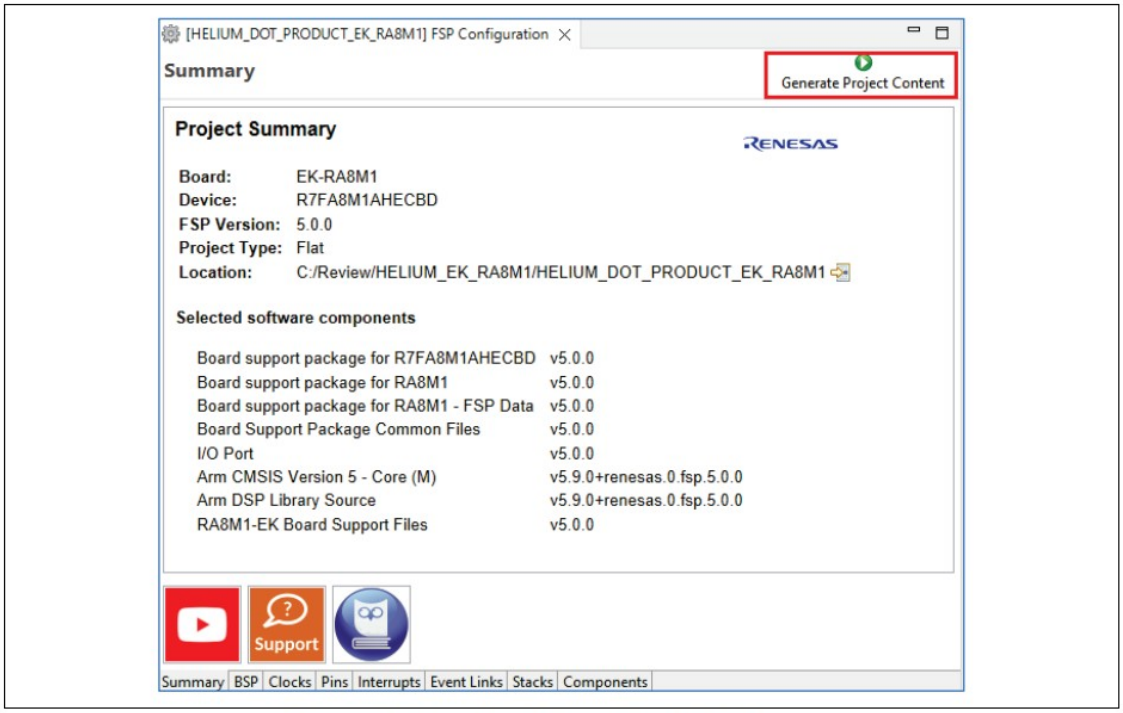


Figure 41. Example of Generating Project Content

Build the active project by selecting Project > Make or Project > Rebuild All .

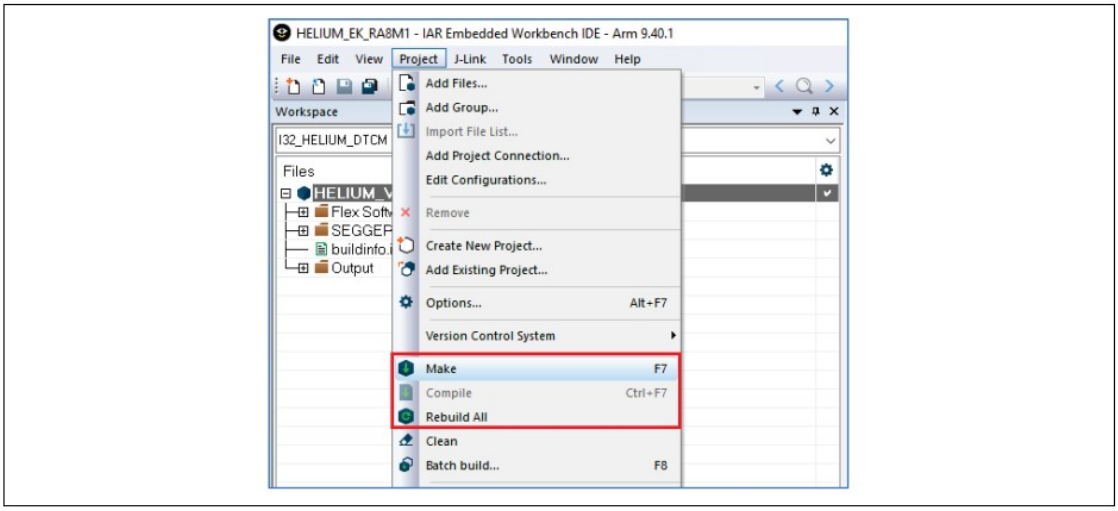


Figure 42. Build the Active Project

5.3 Download and Run Project

The EK-RA8M1 kit has a few switch settings that must be configured before running the projects associated with this application note. These switches must be returned to the default settings per the EK-RA8M1 user manual. In addition to these switch settings, the board also contains a USB debug port and connectors to access the J-Link programming interface.

Table 1. Switch Settings for EK-RA8M1

Switch	Setting
J8	Jumper on pins 1-2
J9	Open

Connect J10 on EK-RA8M1 kit to USB port on your PC, open and start SEGGER RTT Viewer with the following settings.

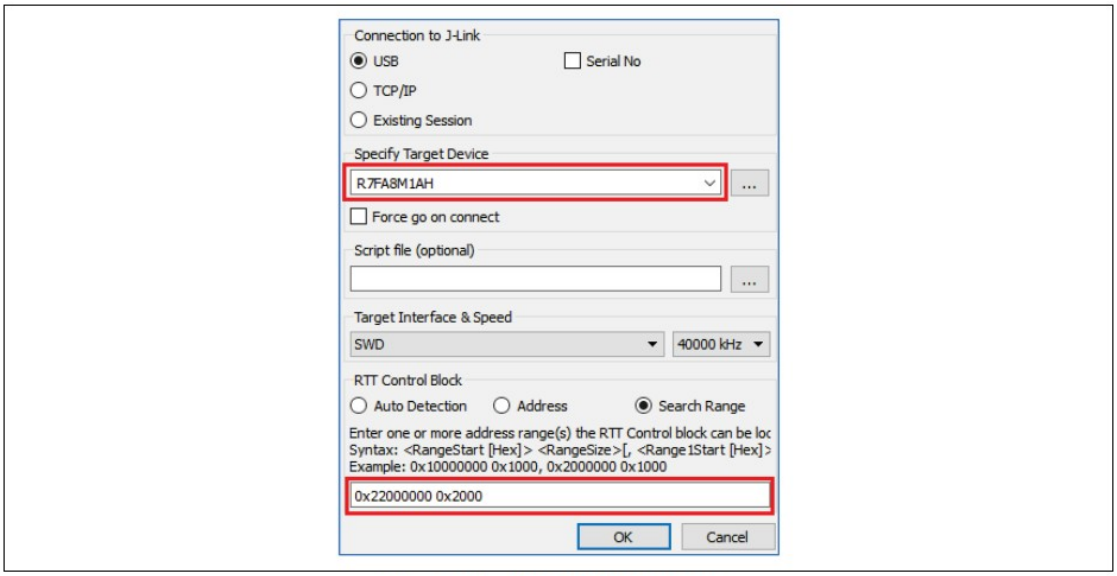


Figure 43. SEGGER RTT Viewer

Click Download and Debug to start running the project.

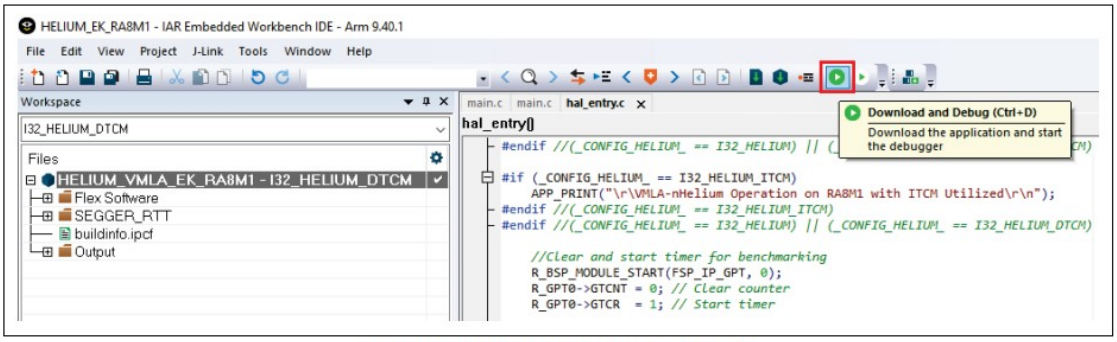


Figure 44. Start Running the Project

The operation results will be printed on SEGGER RTT Viewer, as shown in Figure 45.

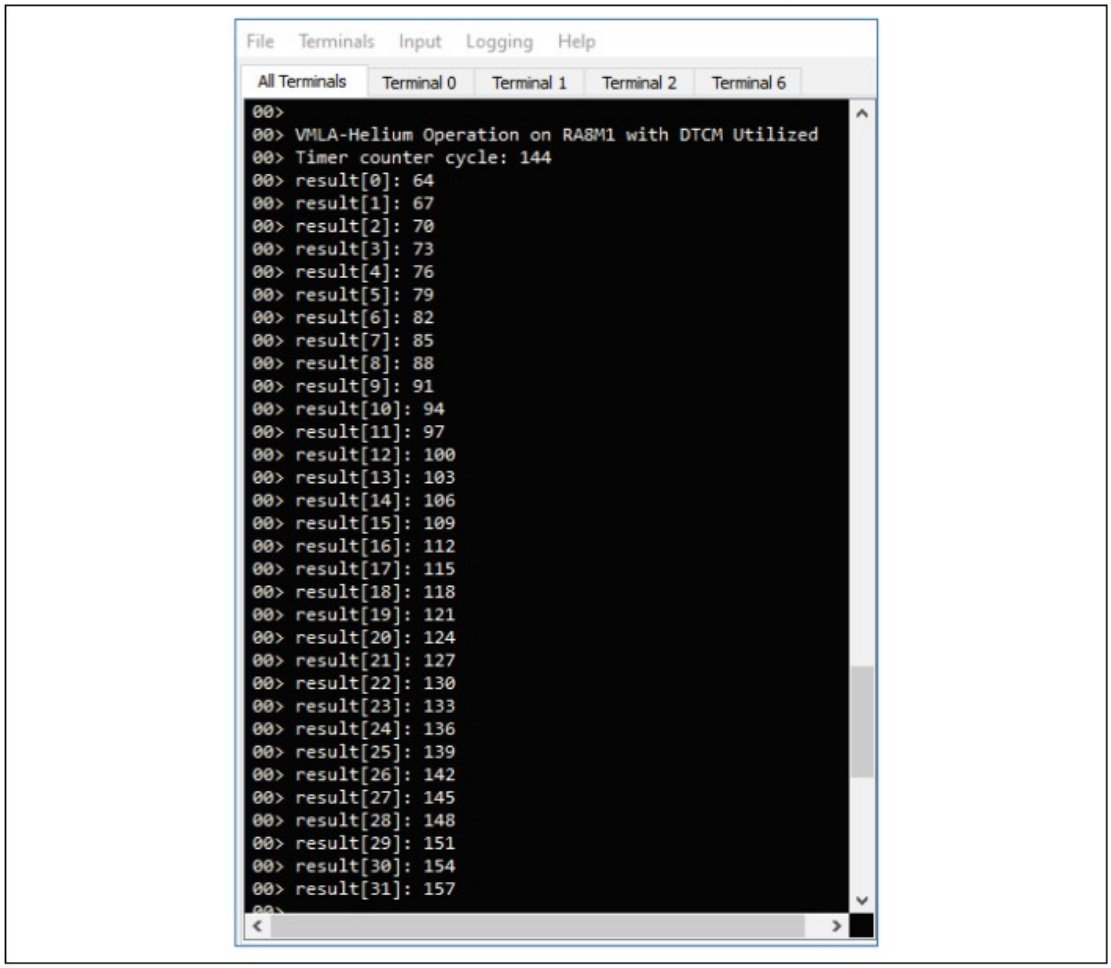


Figure 45. A Helium Operation with DTCM Utilized

5.4 Confirm Instructions Generated For Helium™ Extension

Use the Disassembly window of EWARM to check the Helium™ extension code generated by IAR EWARM compiler.

Figure 46 shows the disassembly of scalar code.

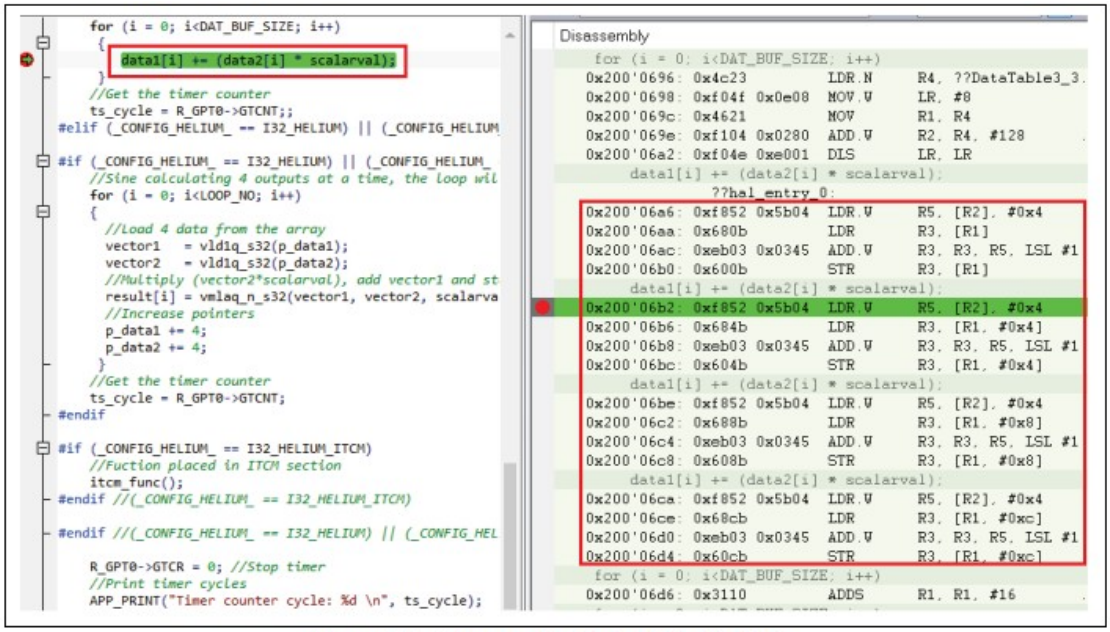


Figure 46. Disassembly Code of Scalar Code

Figure 47 shows the disassembly of Helium code generated using the Helium™ extension.

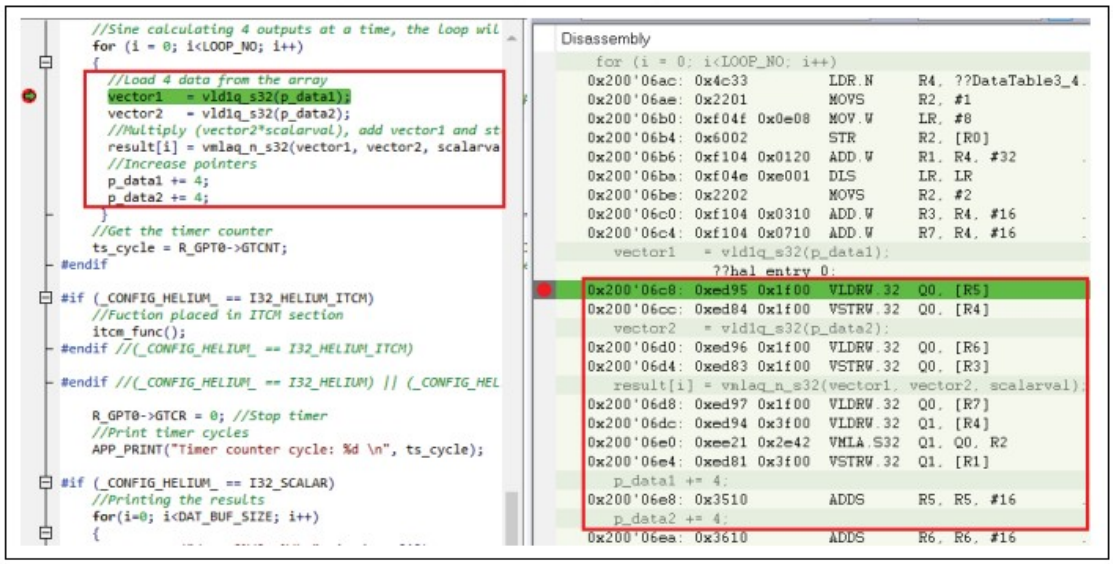


Figure 47. Disassembly of Helium Code Generated by IAR WARM

5.5 Benchmarking Performance

Use the “Timer counter cycle” printed on SEGGER RTT Viewer for performance benchmarking. It shows how many GPT0 counter cycles have elapsed since the function was executed.

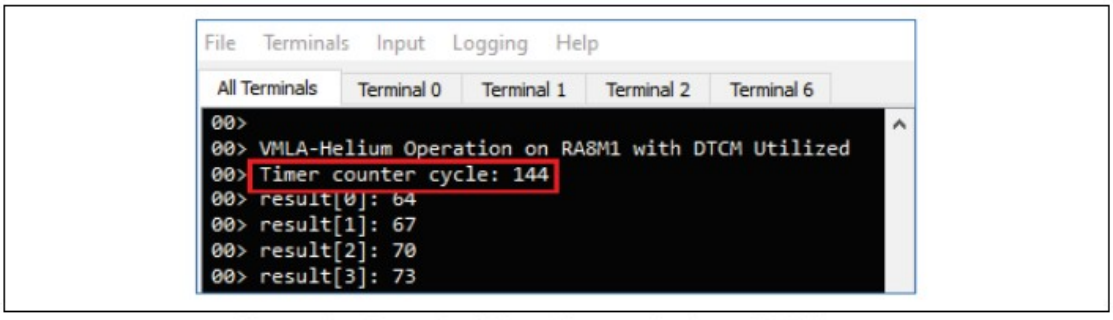


Figure 48. Example of Timer Counter Cycle on RTT Viewer

5.5.1 VMLAVADA Project HELIUM_VMLADAVA_EK_RA8M1

The performances of the function vmladavaq_s32 in various configurations are as follows.

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR	386	
I32_HELIUM	222	42.5
I32_HELIUM_DTCM	148	61.7
I32_HELIUM_ITCM	218	43.5

Figure 49. Performance Data w/o Data Cache Enable

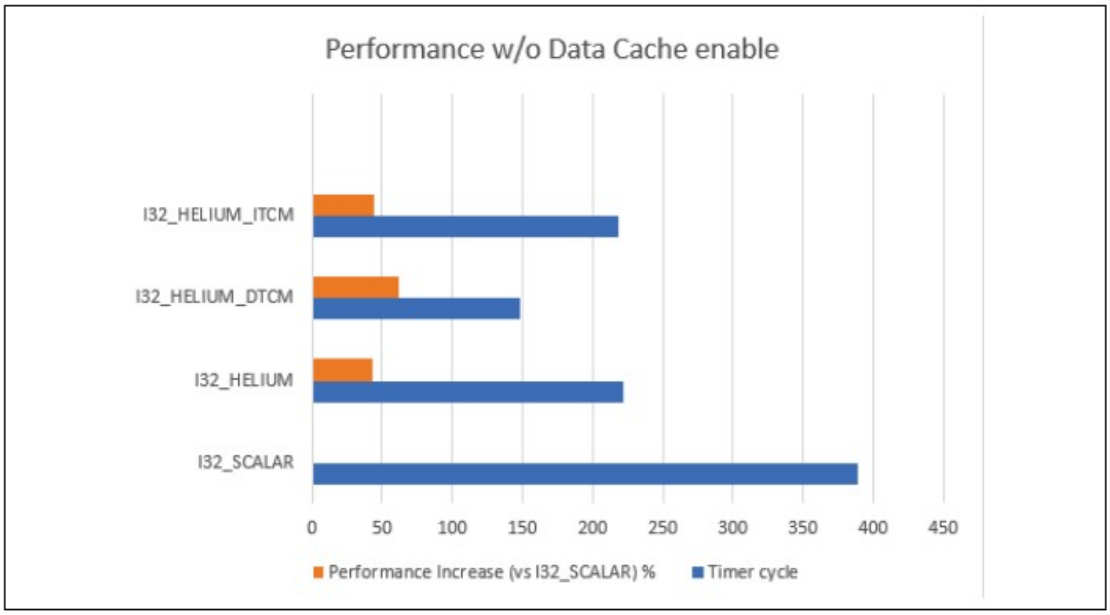


Figure 50. Performance Chart w/o Data Cache Enable

Following are the performances of the vmlaq_n_s32 function with data cache enabled in various configurations. To enable data cache in the project, follows steps in section 4.5, build and download it .

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR (w/o data cache enable)	386	
I32_SCALAR	82	78.8
I32_HELIUM	58	85.0
I32_HELIUM_DTCM	60	84.5
I32_HELIUM_ITCM	52	86.5

Figure 51. Performance Data w/ Data Cache Enable

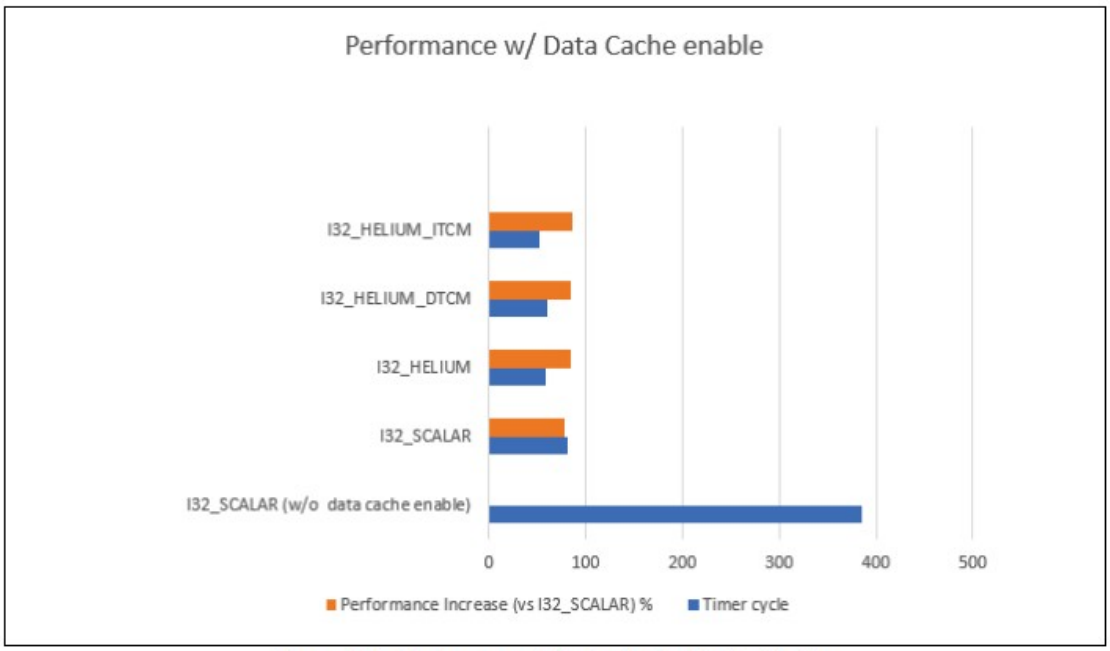


Figure 52. Performance Chart w/ Data Cache Enable

5.5.2 VMLA Project HELIUM_VMLA_EK_RA8M1

The performances of the function vmlaq_n_s32 in various configurations are as follows.

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR	263	
I32_HELIUM	218	17.1
I32_HELIUM_DTCM	144	45.2
I32_HELIUM_ITCM	217	17.5

Figure 53. Performance Data w/o Data Cache Enable

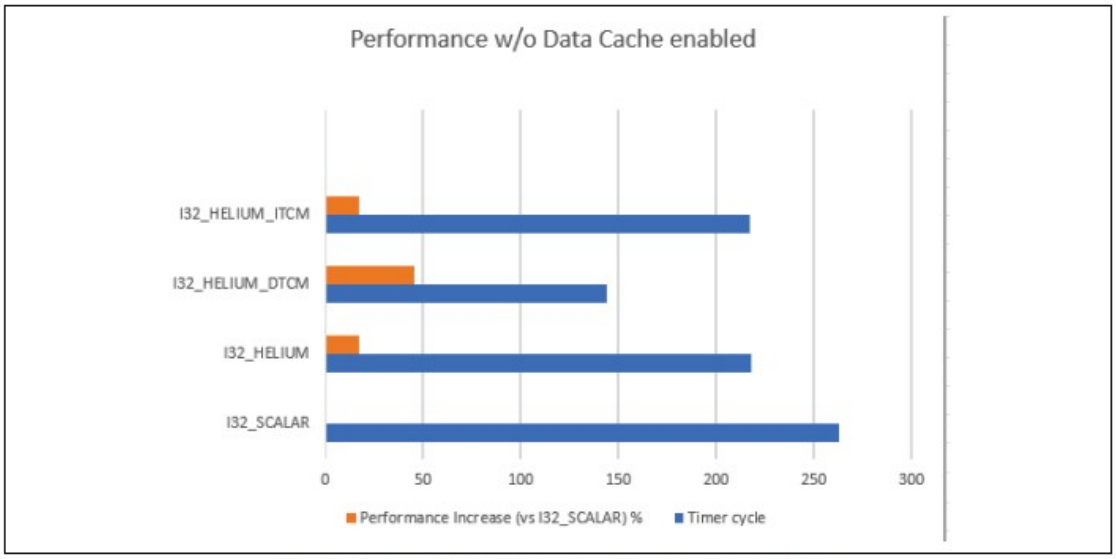


Figure 54. Performance Chart w/o Data Cache Enable

Below are the performances of the vmladavaq_s32 function with data cache enabled in various configurations. To enable data cache in the project, follows steps in section 4.5, build and download it .

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR (w/o data cache enabled)	263	
I32_SCALAR	52	80.2
I32_HELIUM	58	77.9
I32_HELIUM_DTCM	57	78.3
I32_HELIUM_ITCM	52	80.2

Figure 55. Performance Data w/ Data Cache Enable

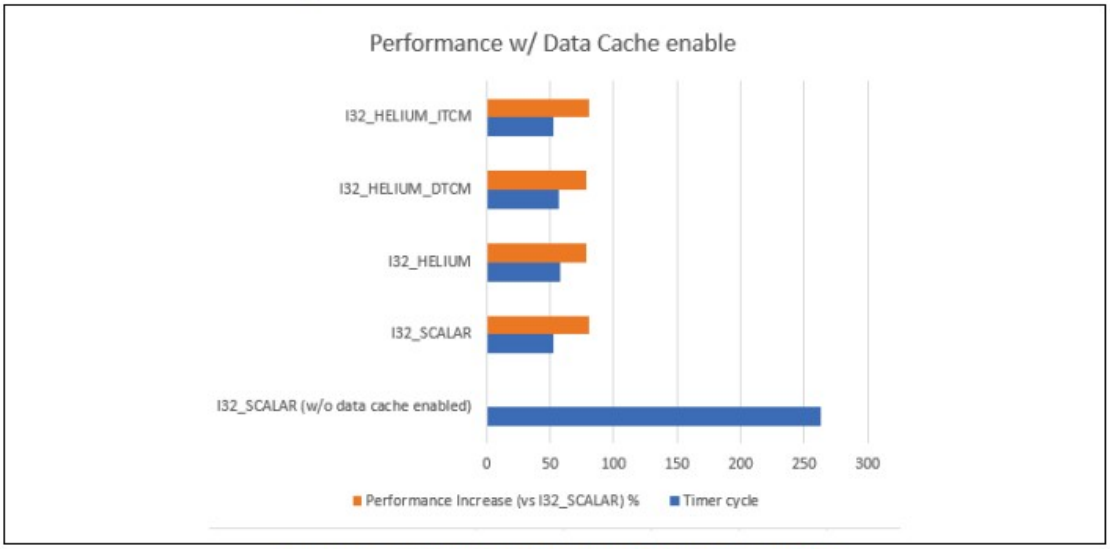


Figure 56. Performance Chart w/ Data Cache Enable

5.5.3 DSP Dot Product Project HELIUM_DOT_PRODUCT_EK_RA8M1

The performances of the ARM DSP Dot Product arm_dot_prod_f32 function in various configurations are as follows.

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR	270	
I32_HELIUM	158	41.5
I32_HELIUM_DTCM	100	63.0

Figure 57. Performance Data w/o Data Cache Enable

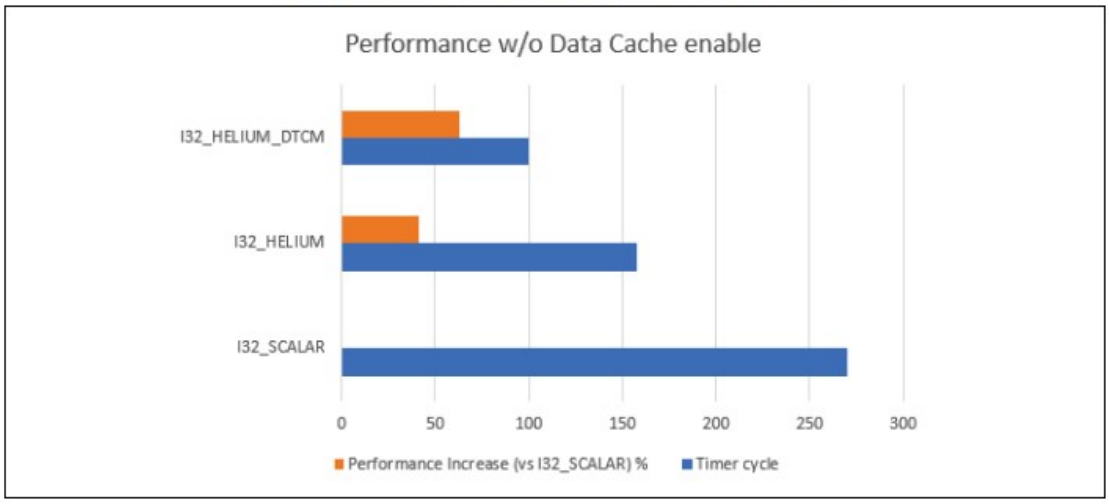


Figure 58. Performance Chart w/o Data Cache Enable

Below are the performances of the ARM Dot Product arm_dot_prod_f32 function with data cache enabled in various configurations. To enable data cache in the project, follows steps in section 4.5, build and download it .

Project Configuration	Timer cycle	Performance Increase (vs I32_SCALAR) %
I32_SCALAR (w/o data cache enable)	270	
I32_SCALAR	49	81.9
I32_HELIUM	36	86.7
I32_HELIUM_DTCM	21	92.2

Figure 59. Performance Data w/ Data Cache Enable

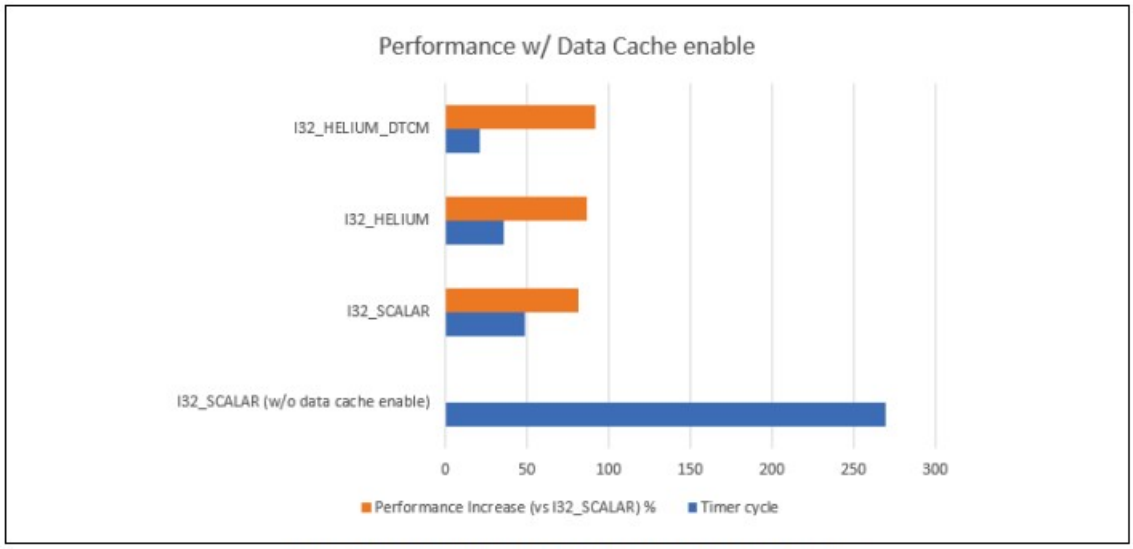


Figure 60. Performance Chart w/ Data Cache Enable

Conclusion

The Renesas RA8 MCU with Arm Cortex-M85 supports significant scalar performance uplift. Furthermore, the Tightly Coupled Memory (TCM) support in Renesas FSP makes it easier to utilize Helium intrinsics and TCM for further improvement.

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information **renesas.com/ra**
RA Product Support Forum **renesas.com/ra/forum**
RA Flexible Software Package **renesas.com/FSP**
Renesas Support **renesas.com/support**

Revision History

Rev.	Date	Description	
		Page	Summary
1.0	Oct.25.23	–	Initial version

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or

losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

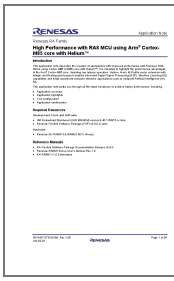
    (Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

    (Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Documents / Resources

	RENESAS RA8 MCU High Performance [pdf] User Guide RA8 MCU High Performance, RA8, MCU High Performance, High Performance, Performance

References

- Flexible Software Package (FSP) | Renesas
- RA ARM Cortex-M 32-bit MCUs, Cortex-M33, M23 and M4 | Renesas
- Renesas Engineering Community
- Renesas Electronics Corporation
- User Manual