AN14600

Replicate the Power Consumption Data of the Data Sheet on the MIMXRT700-EVK Board

Rev. 2.0 — 10 November 2025 Application note

Document information

Information	Content
Keywords	AN14600, i.MX RT700, MIMXRT700-EVK, power consumption
Abstract	This application note describes the procedure to reproduce the power consumption data of the data sheet on the MIMXRT700-EVK board.

1 Introduction

The i.MX RT700 has nine separate power domains and multiple distinct power rails. The chip includes separate compute and sense processing domains that can independently enter or exit Sleep, and Deep-sleep modes. To meet the performance requirements of a system application, the power domains can be active or placed in a low-power state. This power architecture allows for a more granular distribution of power, ensuring allocation when and where it is needed. This document describes the procedure to reproduce the power consumption data of the data sheet on the MIMXRT700-EVK board. It does not include the status of the power domains in various Power modes.

To replicate the power numbers described in this document for the i.MX RT700, perform the procedures explained in the following sections with Software Development Kit (SDK). This document helps users to understand the power consumption performance of the i.MX RT700.

2 Replicate the power numbers

To download the latest SDK for the i.MX RT700, see MCUXpresso SDK Builder. To measure the power consumption data, use power-related examples. Figure 1 shows the SDK version 25.09.00 as an example. If available, use the latest version of the SDK.

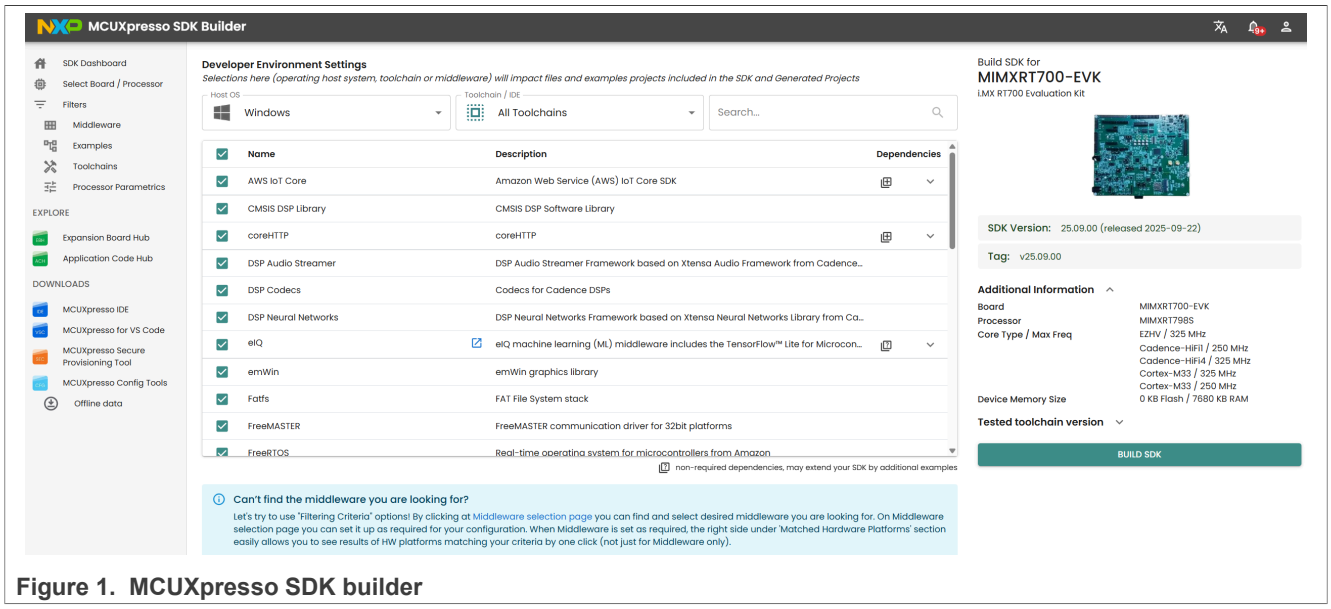

Figure 1. MCUXpresso SDK builder

Figure 2 shows the following three power mode demonstration projects:

- power_mode_comp_only: It is designed for users who require only to use CPU0. In this project, CPU1 enters in the Deep-sleep mode at the beginning.
- power_mode_switch: It shows how CPU0 and CPU1 switch between different power modes.
- power_mode_with_hifi: It shows the power consumption of HIFI4, HIFI1 when running FFT, and the power consumption reproduction of HIFI4.

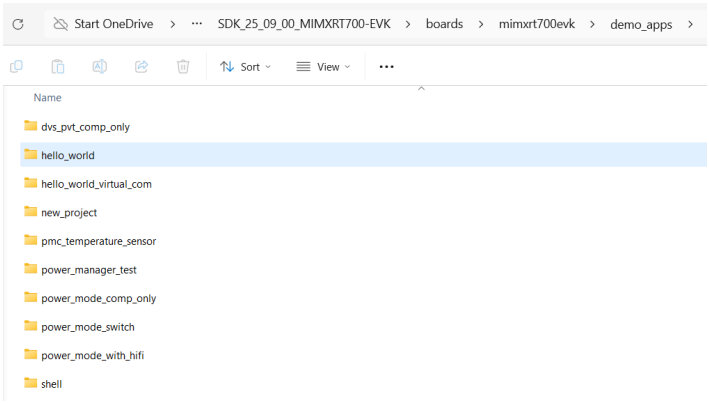

Figure 2. Power mode demonstration projects

Table 1 summarizes all the power cases shown in the *i.MX RT700 Crossover Microcontroller Data Sheet* (document IMXRT700EC) and the projects in which each case is measured. In this document, all the power consumption data is measured based on the IAR projects. To replicate the power numbers use IAR projects and “debug_target”.

Note: *The power_mode_switch project demonstrates all the power modes and the power_mode_comp_only is its subset. For example, in the “dual Deep-sleep mode” case, the projects power_mode_comp_only and the power_mode_switch replicate the power consumption data.*

Table 1. i.MX RT700 power cases in the data sheet

RT700 power cases in the data sheet			
Power demo example projects	**Case**	**HiFi4**	**HiFi1**
power_mode_comp_only	CPU0 CM33 CoreMark, Sense Deep Sleep	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_comp_only	CPU0 CM33 Sleep, Sense Deep Sleep	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_switch	Compute DSR, CPU1 CM33 CoreMark	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_switch	Compute DSR, CPU1 CM33 Sleep	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_with_hifi	CPU0 CM33 Sleep, Sense Deep Sleep, HiFi4 DSP FFT, HiFi1 DSP stalled	HiFi4 Run FFT	HiFi1 DSP stalled
power_mode_with_hifi	Compute DSR, CPU1 CM33 Sleep, HiFi4 DSP power down, HiFi1 DSP FFT	HiFi4 DSP power down	HiFi1 Run FFT
power_mode_comp_only	Compute and sense dual Deep-sleep mode	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_comp_only	Compute and sense dual Deep Sleep Async	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_switch	Full DSR mode	HiFi4 DSP power down	HiFi1 DSP stalled
power_mode_comp_only	Deep-power-down mode	HiFi4 DSP power down	HiFi1 DSP power down
power_mode_comp_only	Full Deep-power-down mode	HiFi4 DSP power down	HiFi1 DSP power down

Before replicating the power consumption data, it must be emphasized that the power numbers of the different chips are different. The actual measurement results are based on the measured chips that you have. The measurement results in this document are only for reference, and the data in the data sheet must be regarded as the standard.

2.1 Replicate power numbers using power_mode_comp_only

This section details the use cases to replicate the power numbers using power_mode_comp_only.

2.1.1 Case - CPU0 CM33 CoreMark, Sense Deep Sleep

Figure 3 shows the case “CPU0 CM33 CoreMark, Sense Deep Sleep” as an example.

Select the following two power consumption data as an example:

- 192 MHz/0.9 V
- 325 MHz/1.1 V

3.1.8.1 CPU0 CM33 CoreMark, Sense Deep Sleep

The table below provides typical current numbers where CPU0 in the compute domain is running CoreMark and the sense domain is in Deep Sleep.

COMPUTE_MAIN_CLK is sourced from FRO0 running at the frequency indicated for each row. HiFi 4 DSP is powered down, and HiFi 1 is stalled.

Current is measured using an external supply for VDD2, VDD1, and VDDN (internal LDOs disabled). VDD1V8_AO = VDDIO_AO = VDD1V8 = VDDIO_0/2/4/5/6 = VDD1V8_EUSB = VDDA_1V8 = VDD1V8_MIPI = VREFH_ANA18 = 1.8 V. VDD3V3_USB = 3.3 V. VDD1V2_EUSB = 1.2 V. VDDN = 1.1 V. VDD1 = 0.63 V.

Body Bias: VDD2 AFBB, VDD2SRAM NBB, VDD1 RBB, VDD1SRAM RBB, VDDN RBB.

Data and code are located in SRAM partition 8.

Table 18. CPU0 CM33 CoreMark, Sense Deep Sleep

Symbol	Description	Min	Typ	Max	Unit	Condition
IVDD2	VDD2 supply current [1]	—	30.43	—	mA	COMPUTE_MAIN_CLK = 325 MHz, VDD2 = 1.1 V
IVDD2	VDD2 supply current [1]	—	20.85	—	mA	COMPUTE_MAIN_CLK = 250 MHz, VDD2 = 1.0 V
IVDD2	VDD2 supply current [1]	—	14.27	—	mA	COMPUTE_MAIN_CLK = 192 MHz, VDD2 = 0.9 V
IVDD2	VDD2 supply current [1]	—	7.69	—	mA	COMPUTE_MAIN_CLK = 110 MHz, VDD2 = 0.8 V
IVDD2	VDD2 supply current [1]	—	3.15	—	mA	COMPUTE_MAIN_CLK = 45 MHz, VDD2 = 0.7 V

Figure 3. CPU0 CM33 CoreMark, Sense Deep Sleep

1. Ensure that the JP1 and JP3 are installed on the MIMXRT700-EVK board. The external Power Management Integrated Circuit (PMIC) PCA9422 powers up the i.MX RT700.
2. Open the power_mode_comp_only project.
3. Open the secondary project and select "debug_target".
4. Modify **DEMO_POWER_SUPPLY_OPTION**, **DEMO_POWER_CPU1_PRINT_ENABLE**, and **DEMO_POWER_ENABLE_DEBUG** as shown in Figure 4. Compile the secondary project.
5. The different SDK versions can have different default configurations. If you are using the different SDK version, modify the SDK to the configuration shown in Figure 4.

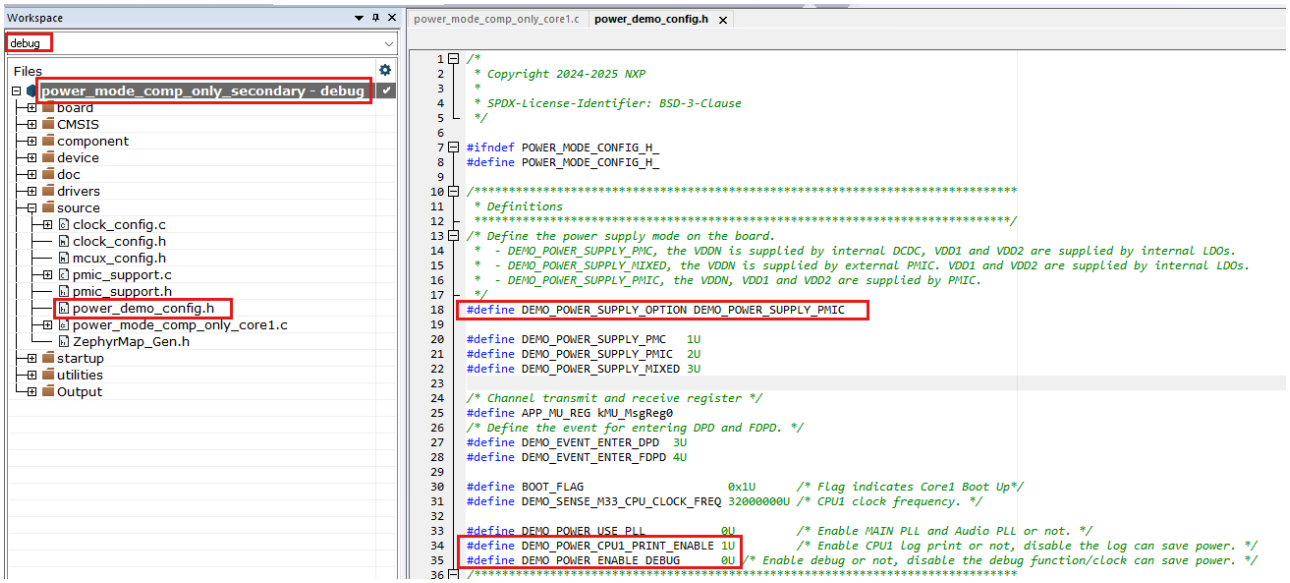

Figure 4. SDK configuration

6. Open the primary project and select "debug_target".
7. Ensure that you do the same changes in `power_demo_config.h` in the primary project, as shown in Figure 4.
8. Add code related to coremark to the primary project, use **coremark_main()** to replace the **SDK_DelayAtLeastUs**, as shown in Figure 5 and Figure 6.
9. Due to the license restrictions, the code related to coremark is not included in the SDK. Port the coremark code. To download the coremark code, see `https://github.com/eembc/coremark`.

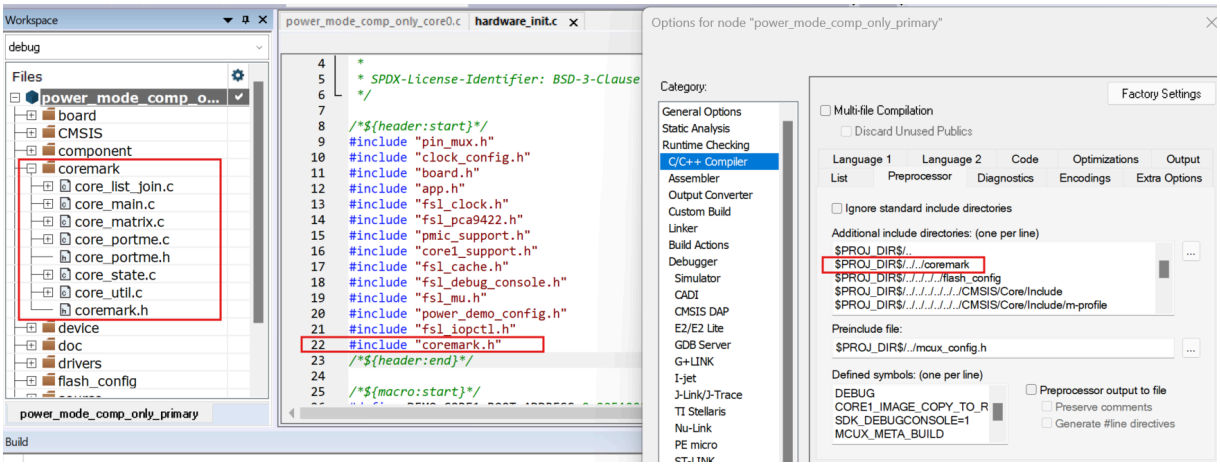

Figure 5. Code related to coremark

power_mode_comp_only_core0.c | hardware_init.c

BOARD_RunActiveTest()

```
530
531  void BOARD_RunActiveTest(void)
532  {
533      uint32_t pinCfg[3] = {0}; /* Used for IOPCTL configuration backup. */
534
535      DEMO_LOG("\r\nThis test mode will keep CPU in run mode but close all other unused modules
536      DEMO_LOG("\r\nPlease don't input any charator until the mode finished.\n");
537
538      /* Deinit unused modules. */
539      BOARD_PMIC_I2C_Deinit();
540      CLOCK_AttachClk(kNONE_to_LPI2C15);
541      DEMO_DeinitDebugConsole();
542      CLOCK_AttachClk(kNONE_to_FCCLK0);
543
544      pinCfg[0] = IOPCTL0->PIO[0][31];
545      pinCfg[1] = IOPCTL0->PIO[1][0];
546      pinCfg[2] = IOPCTL0->PIO[0][9];
547
548      CLOCK_DisableClock(kCLOCK_Rtc);
549      /* Power down unused modules. */
550  #if (DEMO_POWER_USE_PLL == 0U) /* PLL located in VDDN. */
551      if (!IS_XIP_XSPI0() && !IS_XIP_XSPI1())
552      {
553          /* XSPI0 and XSPI1 memory interface located in VDDN_COM. */
554          POWER_EnablePD(kPDRUNCFG_DSR_VDDN_COM);
555          POWER_EnableRunRBB(kPower_BodyBiasVddn);
556          POWER_EnablePD(kPDRUNCFG_SHUT_COMNN_MAINCLK);
557          POWER_ApplyPD();
558      }
559  #endif
560      POWER_EnablePD(kPDRUNCFG_APD_OCOTP);
561      POWER_ApplyPD();
562
563      BOARD_DisableClocks();
564
565      /* Note, the debug will not work anymore when the sense shared mainclk is disabled. */
566      POWER_EnablePD(kPDRUNCFG_PD_LPOSC);
567      POWER_EnablePD(kPDRUNCFG_SHUT_SENSES_MAINCLK);
568
569      CLOCK_DisableClock(kCLOCK_Sleepcon0);
570
571      /* Simulate a task. */
572      uint32_t i;
573      for (i = 0U; i < 500U; i++)
574      {
575          coremark_main();
576      }
577
578      /* Restore clock, power for used modules. */
579      CLOCK_EnableClock(kCLOCK_Sleepcon0);
```

Figure 6. coremark_main

10. Ensure that the link file of the primary project is changed to let CPU0 code or data both run at sram partition8. If the SDK has already done it, ignore it.

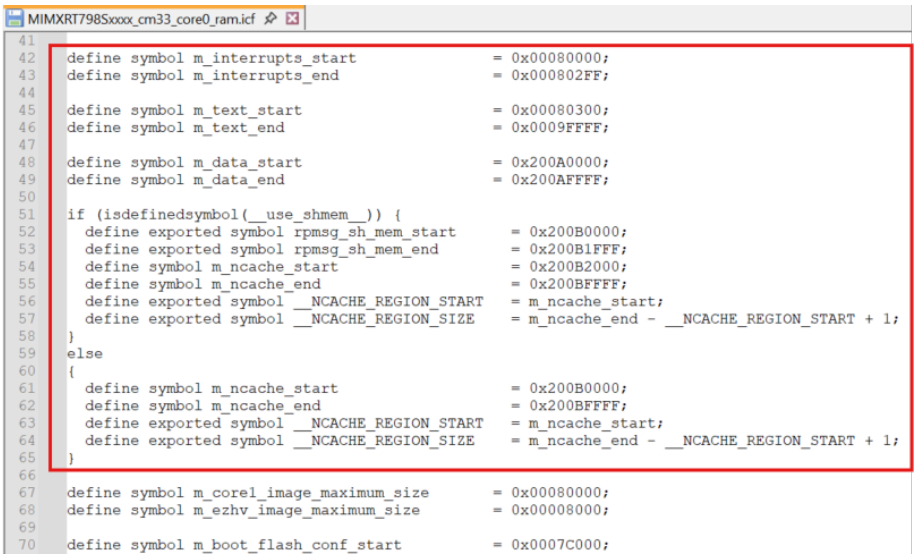

Figure 7. Link file

11. Compile to generate the binary file **power_mode_comp_only_primary.bin**. Figure 8 shows the binary file.
12. Download the image to the address **0x28000000**. Figure 9 shows an example of downloading an image to the address **0x28000000** with JLINK. To download the binary file to flash, you can choose other methods such as blhost.

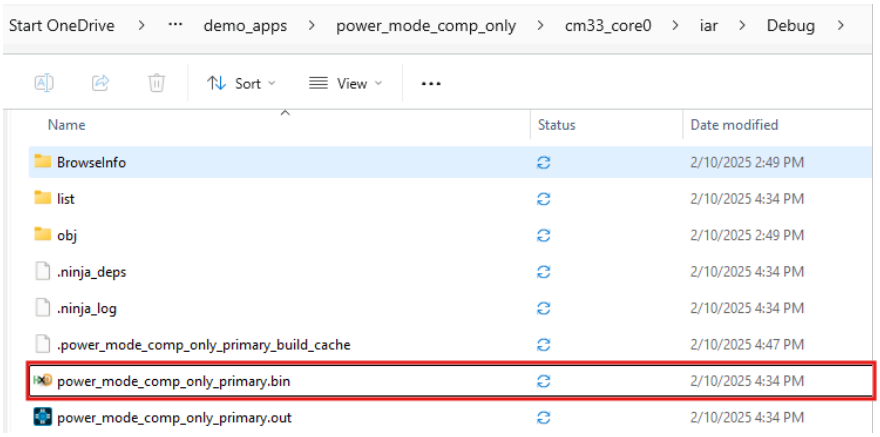

Figure 8. Generate the binary file

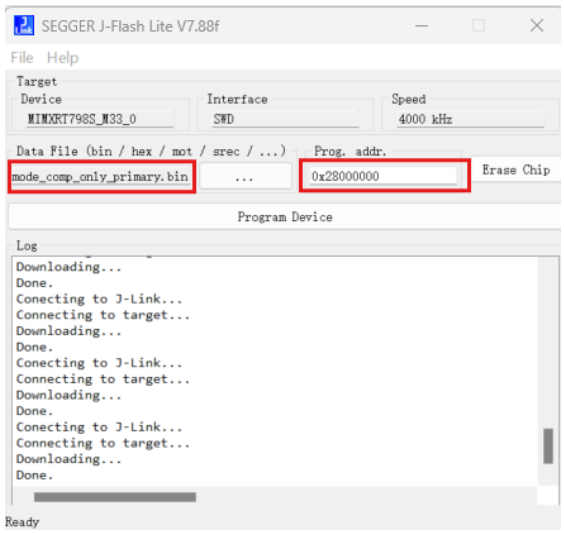

Figure 9. Download the image

13. After the demo boots successfully, see the log from the CPU0 com port. For more details, see Figure 10. The default setting of CPU0 is 192 MHz/0.9 V. To measure the power consumption data and JP1 current on the MIMXRT700-EVK board, select the option 6 **Active test mode**. Figure 10 shows the power number of VDD2. For power measurements, you could use the Joulescope or digital multimeter.
When measuring the current, a multimeter is connected in series with the circuit, which causes a certain voltage drop and triggers the LVD. Therefore, it is necessary to configure the LVD. In the "hardware_init.c", find “POWER_SelectRunSetpoint(kRegulator_Vdd2LDO, 2U)” and change the value from "2" to "0".

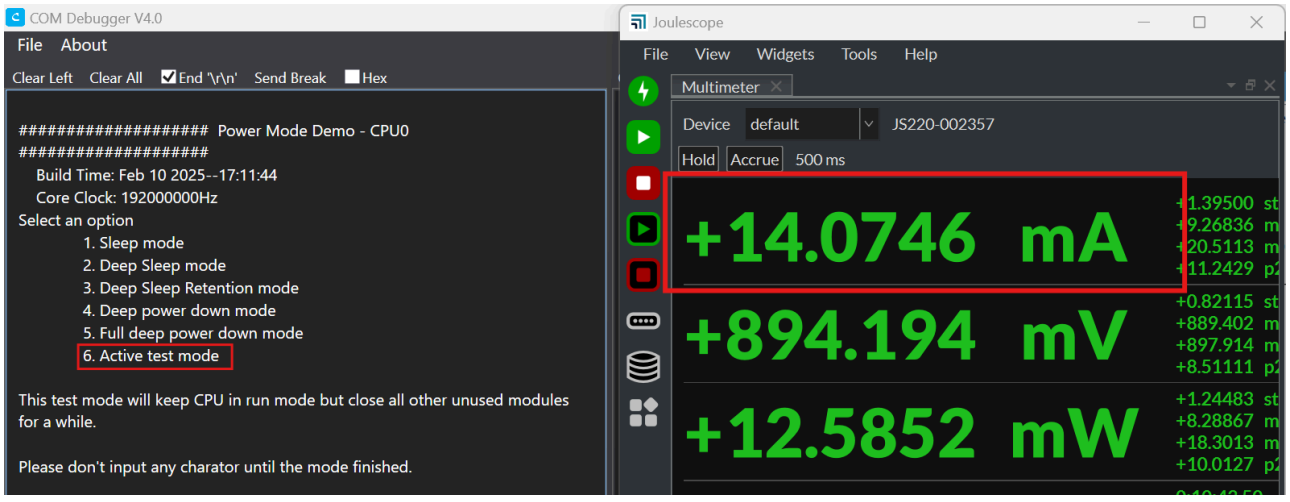

Figure 10. Log from the CPU0 com port and the power number of VDD2

14. Continue to measure the power consumption at different voltages and frequencies. Set the CPU0 to 325 MHz/1.1 V. Figure 11 shows the changed CPU0 frequency from 192 MHz to 325 MHz.

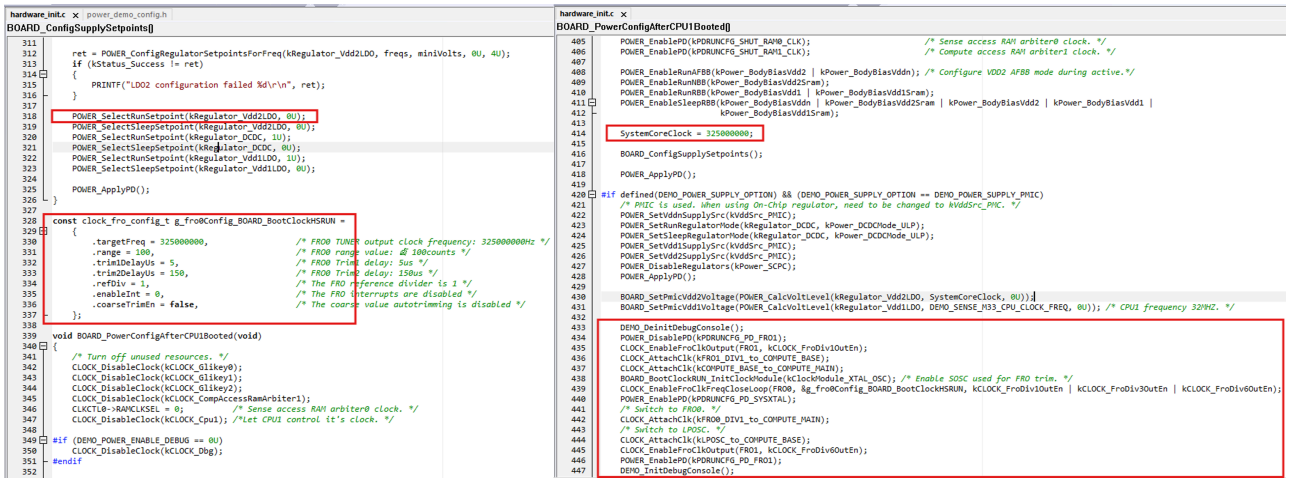

Figure 11. Changed CPU0 frequency 325 MHz

Repeat the step 6 and step 7. Figure 12 shows the power number of VDD2.

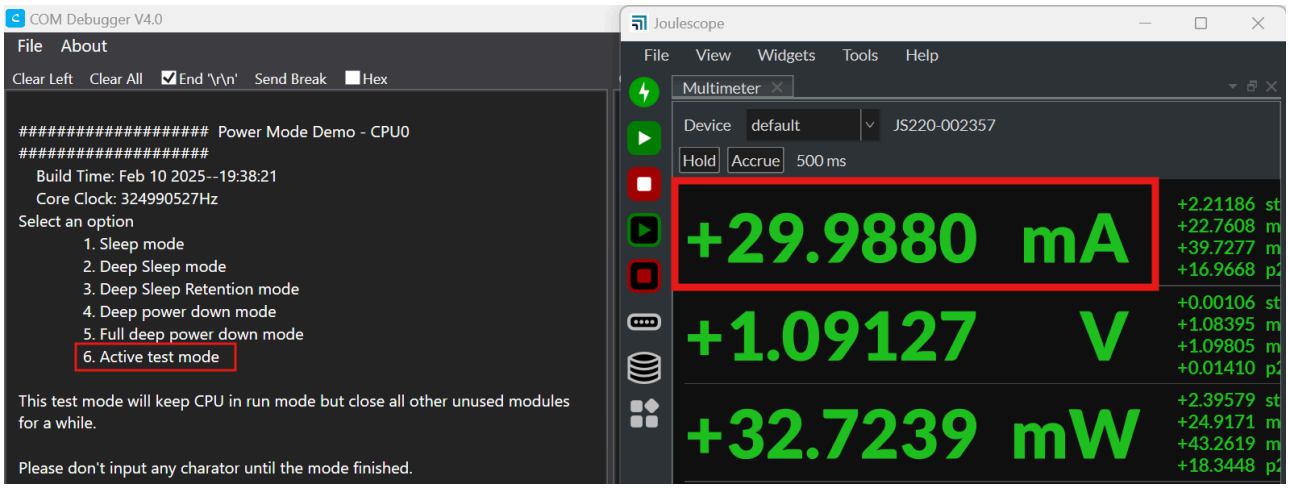

Figure 12. Power number of VDD2

15. Voltage compensation: It must be emphasized that the impedance of the ammeter causes the voltage drop. If you must get the current at a more accurate voltage point, voltage compensation is required. Increase the voltage of VDD2 with one step, and try to make the actual voltage of VDD2 close to 1.1 V. Figure 13 shows the power consumption.

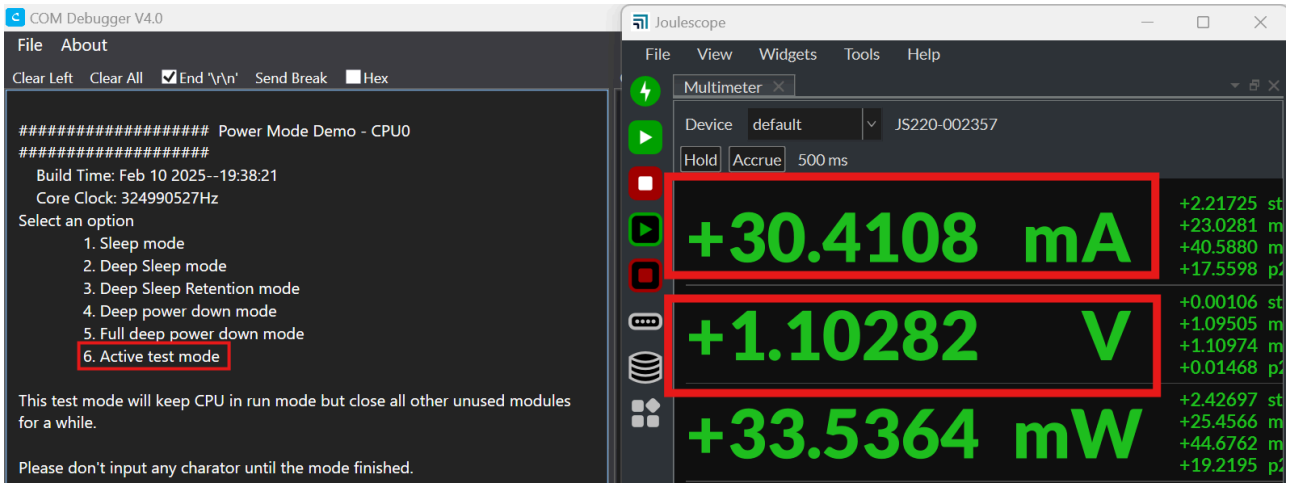

Figure 13. Power consumption

In this case, the power consumption data of 192 MHz/0.9 V and 325 MHz/1.1 V are close to the data in Figure 3.

To modify the frequency and voltage to measure the power consumption data with other frequency and voltage combinations, follow step 14. Then you can get the power consumption data directly.

2.1.2 Case - CPU0 CM33 Sleep, Sense Deep Sleep

For this case, consider CPU0 192 MHz/0.9 V as an example.

Use the modified power_mode_comp_only project and repeat the step 1 to step 15 of Section 2.1.1. In step 13, select option 1 **Sleep mode** instead of option 6 **Active test mode**. Enter **T**, and then select **9**, CPU0 enters in the Sleep mode and the Real-Time Clock (RTC) awakes it after 9 seconds. Figure 14 shows the measured power consumption of VDD2 during this period.

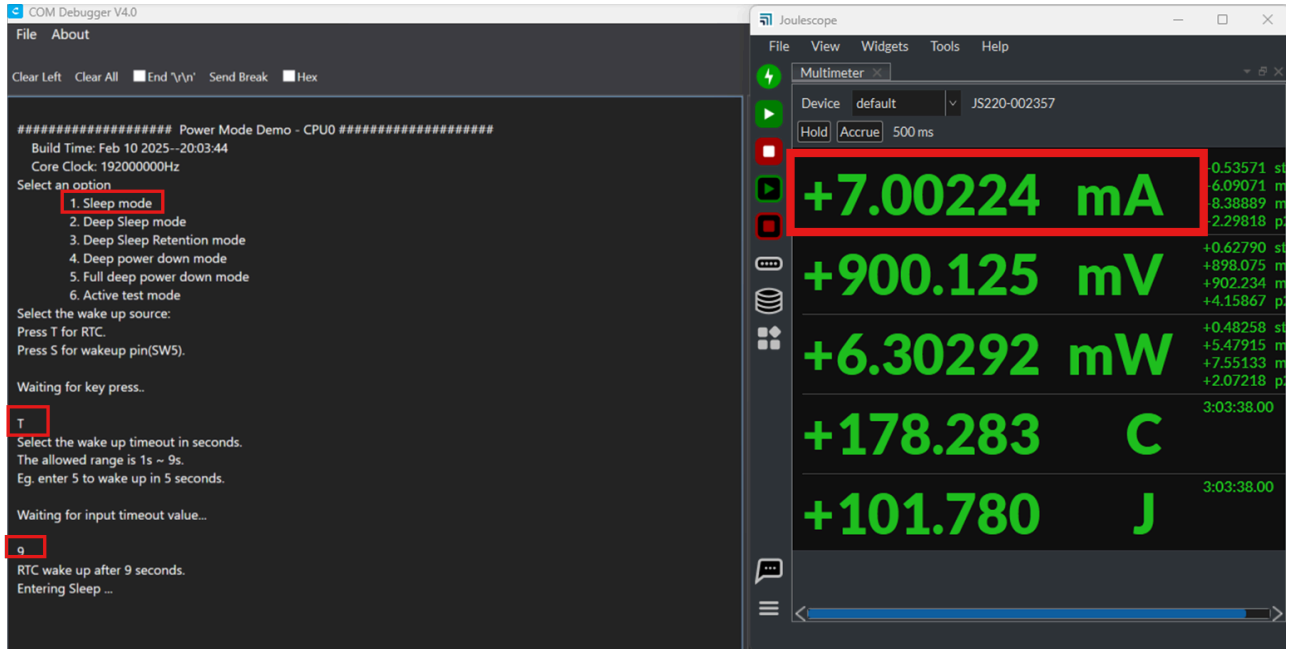

Figure 14. The power consumption of VDD2

2.1.3 Case - Compute and Sense Deep Sleep Async

For this case, repeat the step 1 to step 5 and run the modified power_mode_comp_only project again. Select the option 2 **Deep Sleep mode**. Enter **T**, and then select **9** in CPU0 com. Measure the JP1 current, the VDD2 current is approx. 40.1 µA as shown in Figure 15. Measure the JP3 current, the VDD1 current is approx. 13.7 µA as shown in Figure 16.

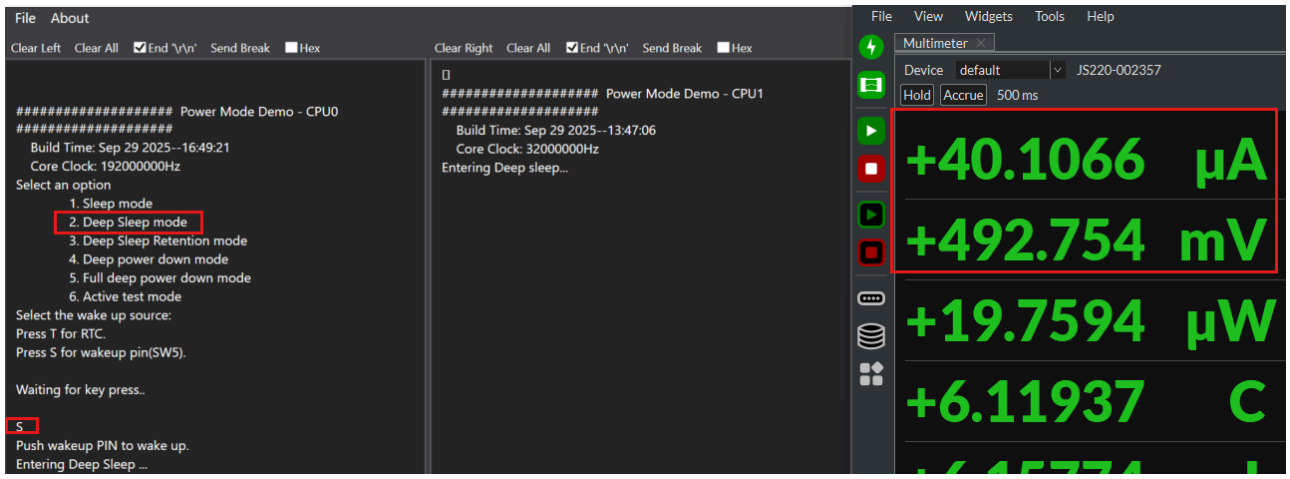

Figure 15. The Deep-sleep mode and measure the JP1 current

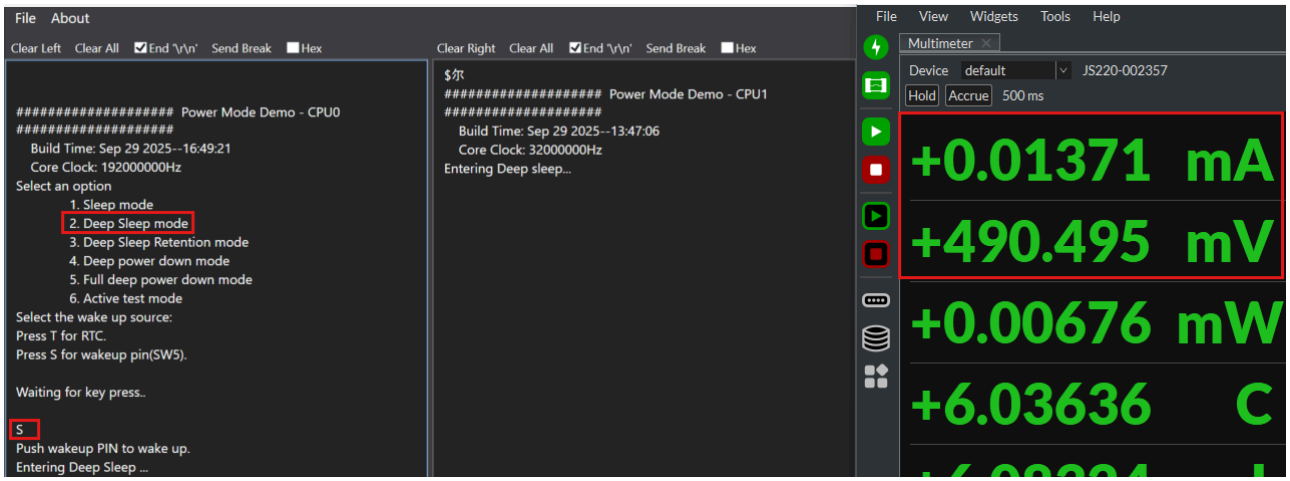

Figure 16. The Deep-sleep async and measure JP3 current

2.1.4 Case - CPU0 CM33 and CPU1 CM33 Deep-power-down mode

For this case, wait for the CPU0 com port to generate the interactive information and then select option 4 **Deep power down mode**.

2.1.5 Case - CPU0 CM33 and CPU1 CM33 full Deep-power-down mode

For this case, wait for the CPU0 com port to generate the interactive information and then select option 5 **Full deep power down mode**.

2.2 Replicate power numbers using power_mode_switch

This section details the use cases to replicate the power numbers using power_mode_switch.

2.2.1 Case - CPU0 CM33 DSR, CPU1 CM33 Coremark

To ensure that the JP1 and JP3 on the MIMXRT700-EVK board are installed, repeat the step 1 of Section 2.1.1. The external PMIC PCA9422RT700 powers up the i.MX RT700.

1. See step 2 to step 5 of Section 2.1.1. Open the power_mode_switch and secondary projects.
2. Select the "debug_target", and modify the **DEMO_POWER_SUPPLY_OPTION**, and **DEMO_POWER_ENABLE_DEBUG** as shown in Figure 17.

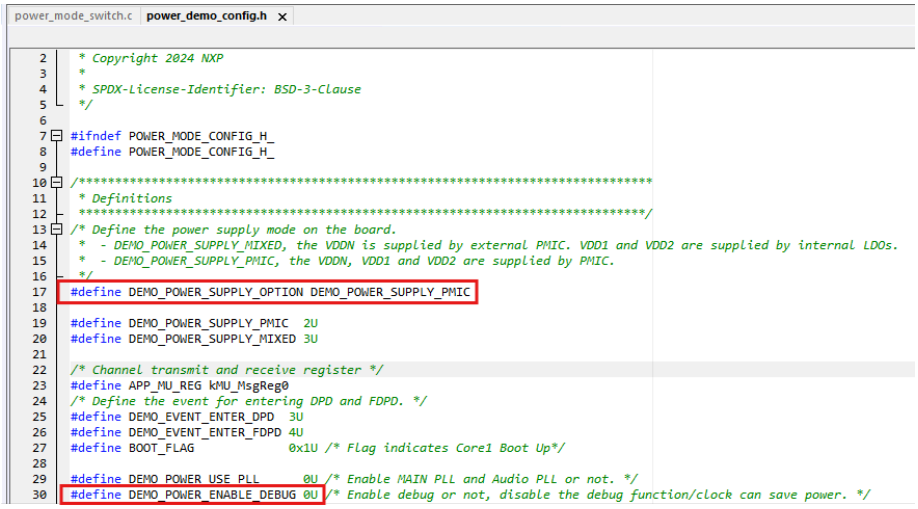

Figure 17. Modify the projects

3. See step 8 and step 9 of Section 2.1.1. Add code related to coremark to the secondary project and use **coremark_main()** to replace the '**SDK_DelayAtLeastUs**' as shown in Figure 18 and Figure 19.

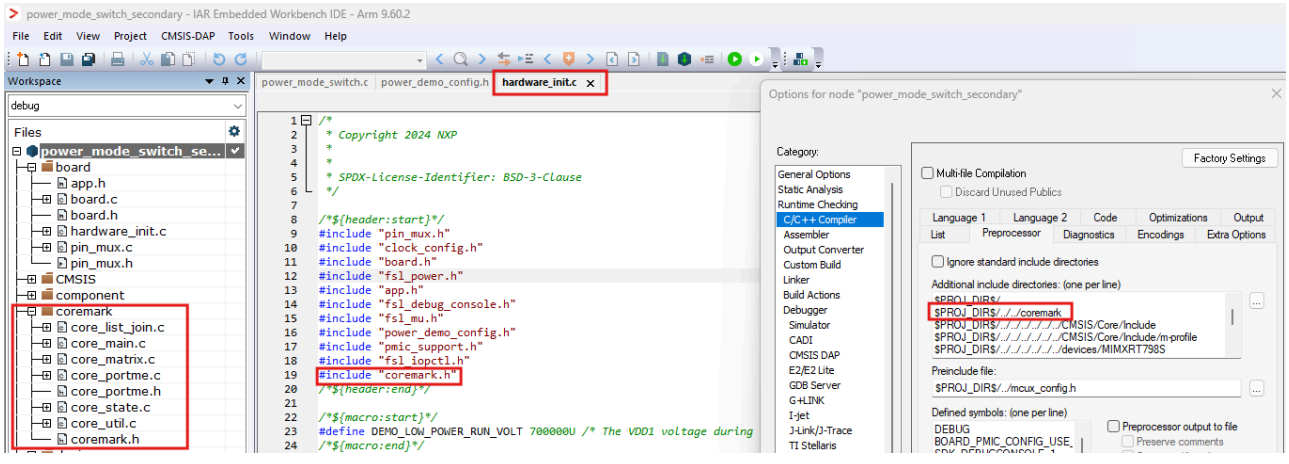

Figure 18. Code related to coremark

power_mode_switch.c | power_demo_config.h | hardware_init.c

```
184  void BOARD_RunActiveTest(void)
185  {
186      DEMO_LOG("\r\nThis test mode will keep CPU in run mode but close all other unused modules for a while.\n");
187      DEMO_LOG("\r\nPlease don't input any charator until the mode finished.\n");
188
189      DbgConsole_Deinit();
190
191      /* Disable clocks - CLKCTL1(Sense private) */
192      CLOCK_DisableClock(kCLOCK_Syscon1); /* CLKCTL1->PSCCTL0 */
193      CLKCTL1->PSCCTL1 = 0U; /* Disable clock for INPUTMUX, WWDT2-3, MU3, SEMA42_3, UTICK1, MRT1, CTIMER5-7, PINT, GPIO,
194                                FLEXCOMM, eDMA, HiFI1, SenseAccessRAM0. */
195
196      /* Disable clocks - CLKCTL3(Sense shared) */
197      CLOCK_DisableClock(kCLOCK_Mu1);
198      CLOCK_DisableClock(kCLOCK_Iopctl1);
199      CLOCK_DisableClock(kCLOCK_Syscon3);
200      CLOCK_DisableClock(kCLOCK_Sema420);
201      CLOCK_DisableClock(kCLOCK_LPI2c15);
202      CLOCK_DisableClock(kCLOCK_Rtc);
203
204      /* Disable clock slice. */
205      CLOCK_AttachClk(kNONE_to_LPI2C15);
206      CLOCK_AttachClk(kNONE_to_FLEXCOMM19);
207
208      CLOCK_EnableFroClkOutput(FRO2, kCLOCK_FroDiv1OutEn);
209
210      POWER_EnablePD(kPDRUNCFG_APD_OCOTP); /* ERR052483 */
211      POWER_ApplyPD();
212      /* Note, the debug will not work anymore when the sense shared mainclk is disabled. */
213      POWER_EnablePD(kPDRUNCFG_PD_LPOSC);
214      POWER_EnablePD(kPDRUNCFG_SHUT_SENSES_MAINCLK);
215      CLOCK_DisableClock(kCLOCK_Sleepcon1);
216
217      /* Simulate a task. */
218      uint32_t i;
219      for (i = 0U; i < 500U; i++)
220      {
221          coremark_main();
222      }
223
224      CLOCK_EnableClock(kCLOCK_Sleepcon1);
225      CLOCK_EnableClock(kCLOCK_Syscon1); /* CLKCTL1->PSCCTL0 */
226      POWER_DisablePD(kPDRUNCFG_PD_LPOSC);
227      POWER_DisablePD(kPDRUNCFG_SHUT_SENSES_MAINCLK);
228      CLOCK_EnableClock(kCLOCK_Iopctl1); /* In CLKCTL3, requires sense shared clock. */
229      CLOCK_EnableFroClkOutput(FRO2, kCLOCK_FroDiv1OutEn | kCLOCK_FroDiv3OutEn | kCLOCK_FroDiv6OutEn);
```

Figure 19. coremark_main

4. See step 10 of Section 2.1.1. Ensure that the link file of the secondary project is changed to let CPU1 code or data both run at sram partition22. If the SDK has already done this step, ignore it. Then compile the secondary project.
5. See step 11, and step 12 of Section 2.1.1. Open the primary project and select "debug_target".
6. Ensure that you do the same changes in `power_demo_config.h` in secondary project, as shown in Figure 17.
7. To generate the binary file **power_mode_switch_primary.bin**, compile the primary project. Figure 20 shows the binary file.
8. Download the image to the address `0x28000000`. Figure 21 shows an example of downloading an image to the address `0x28000000` with JLINK.

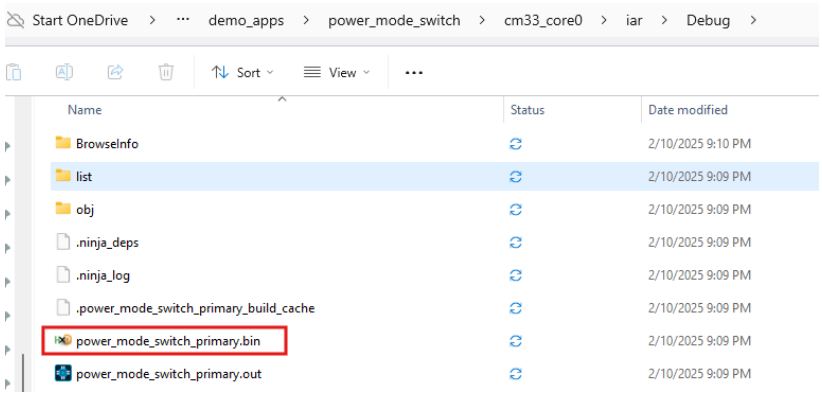

Figure 20. Generate the binary file

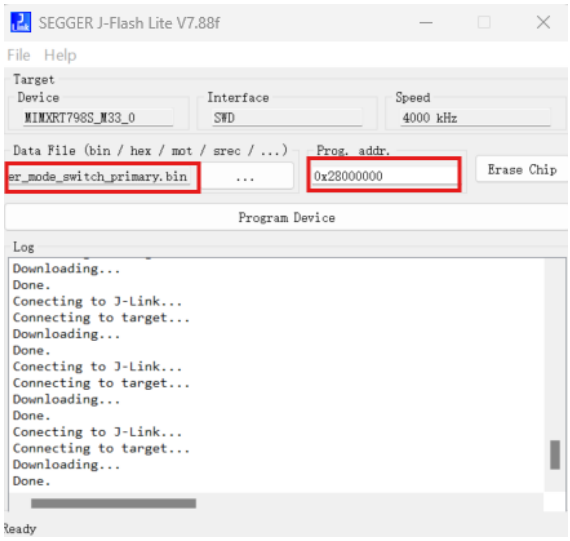

Figure 21. Download the image

9. See step 13 of Section 2.1.1, after the demo boots successfully, see the log from CPU0 com port and CPU1 com port. For more details, see Figure 22. The default setting of the CPU1 is 96 MHz/0.9 V. Take 160 MHz/0.9 V as an example, modify the frequency as shown in Figure 23, and compile the secondary project.

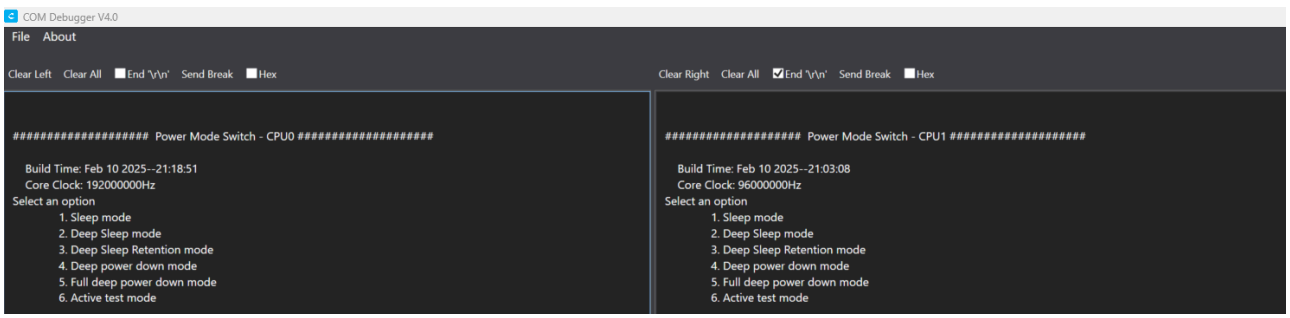

Figure 22. Log from CPU0 and CPU1

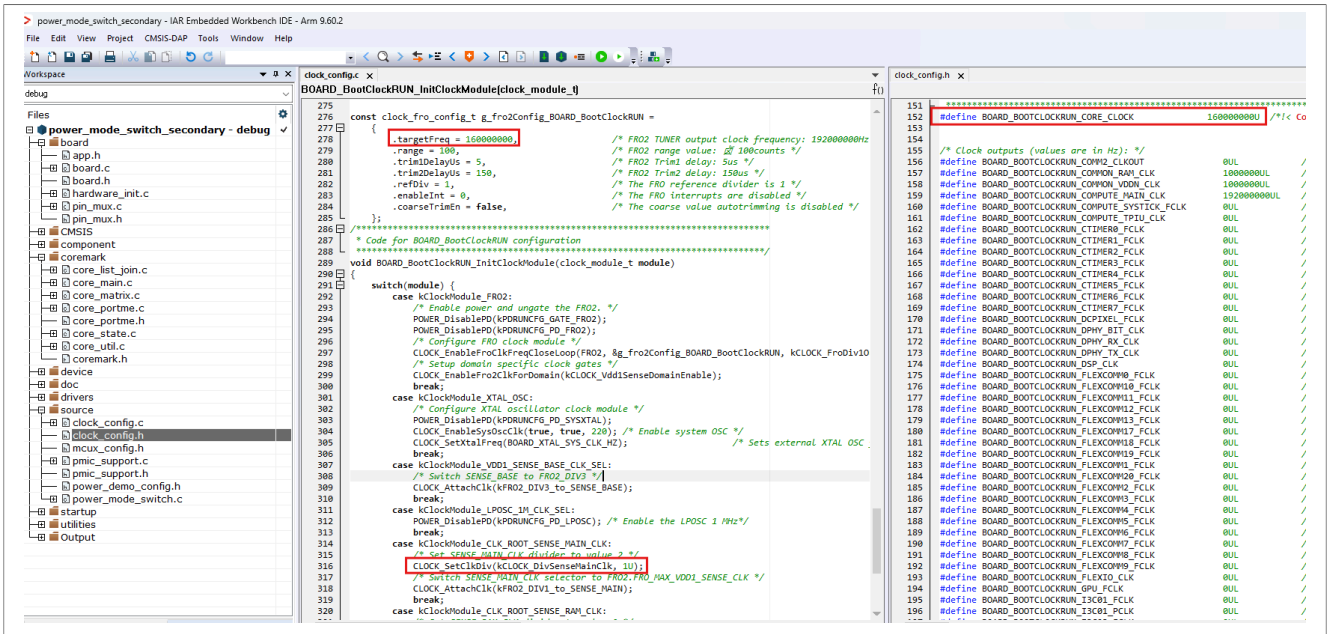

Figure 23. Changed CPU1 frequency 160 MHz

10. See step 11 to step 13 of Section 2.1.1, after the demo boots successfully, see the logs from CPU0 and CPU1 com ports. For more details, see Figure 22. The setting of CPU1 must be 160 MHz/0.9 V.
11. First select option 3 “Deep Sleep Retention mode”, enter “N” in CPU0 com, then select option 6 “Active test mode” in CPU1 com.
12. Measure the JP3 current on the MIMXRT700-EVK board. Figure 24 shows the power number of VDD1. If you need a more accurate measurement, see step 15 of Section 2.1.1.
    When measuring the current, a multimeter is connected in series with the circuit, which causes a certain voltage drop and triggers the LVD. Therefore, it is necessary to configure the LVD. In the "hardware_init.c", find “POWER_SelectRunSetpoint(kRegulator_Vdd1LDO, 2U)” and change the value from "2" to "0".

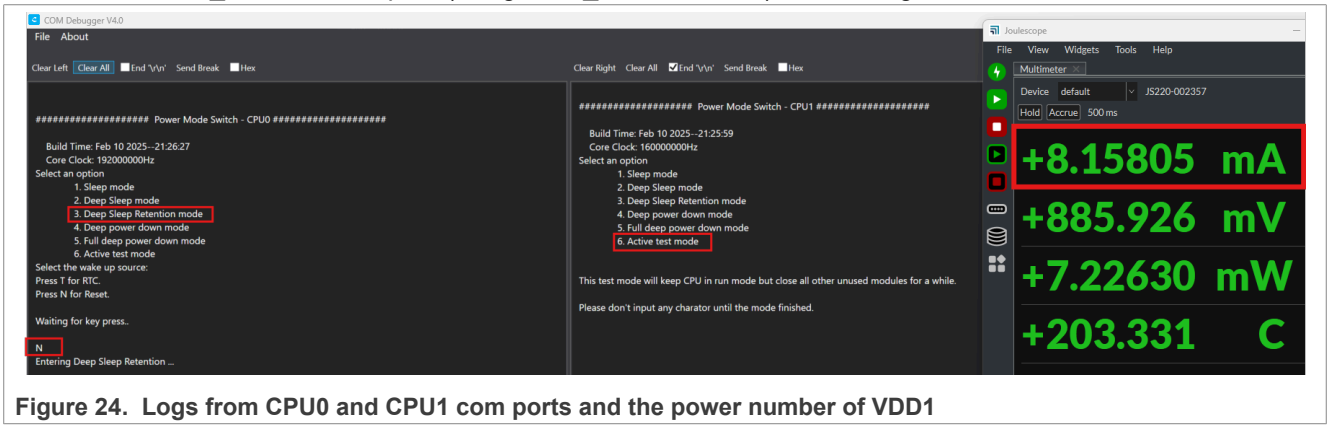

Figure 24. Logs from CPU0 and CPU1 com ports and the power number of VDD1

2.2.2 Case - CPU0 CM33 DSR, CPU1 CM33 Sleep

For this case, consider the CPU1 160 MHz/0.9 V as an example.

Use the modified power_mode_switch project and repeat the step 1 to step 13. In step 13, select option 1 **Sleep mode** instead of option 6 **Active test mode**. Enter **T**, and then select **9**, CPU1 enters in the Sleep mode and RTC awakes it after 9 seconds. Figure 25 shows the measured power consumption of VDD1 during this period.

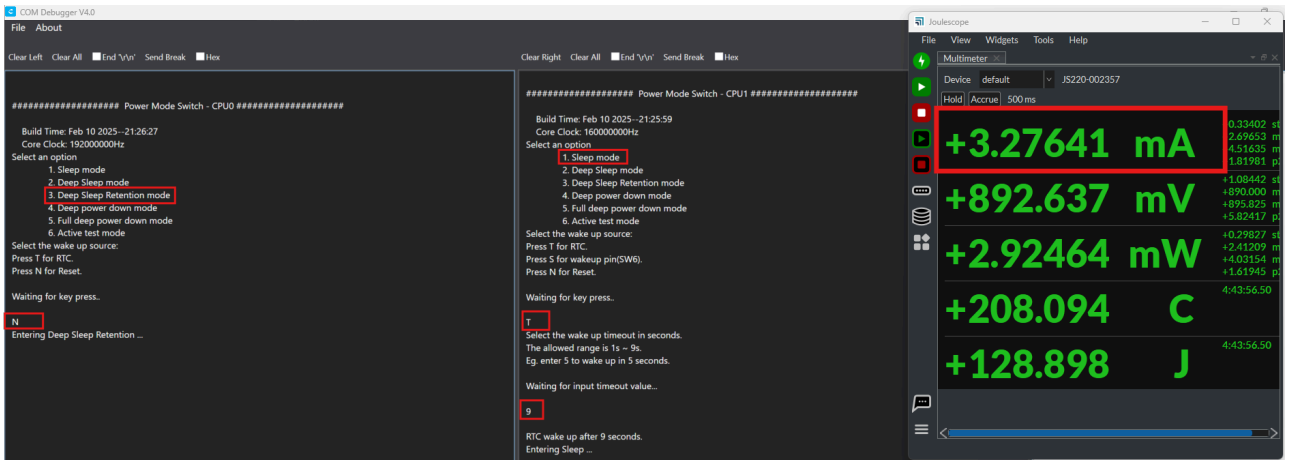

Figure 25. The power consumption of VDD1

2.2.3 Case - CPU0 CM33 DSR, CPU1 CM33 DSR

For this case, repeat the step 1 to step 5. Select option 3 **Deep Sleep Retention mode** and enter **N** in the CPU0 com. Measure the JP1 current, the VDD2 current is approx. 30.8 µA as shown in Figure 26. At the same time, measure the JP3 current and the VDD1 current is approx. 9.7 µA as shown in Figure 27. Ensure that the CPU0 enters the DSR mode before the CPU1 enters the DSR mode.

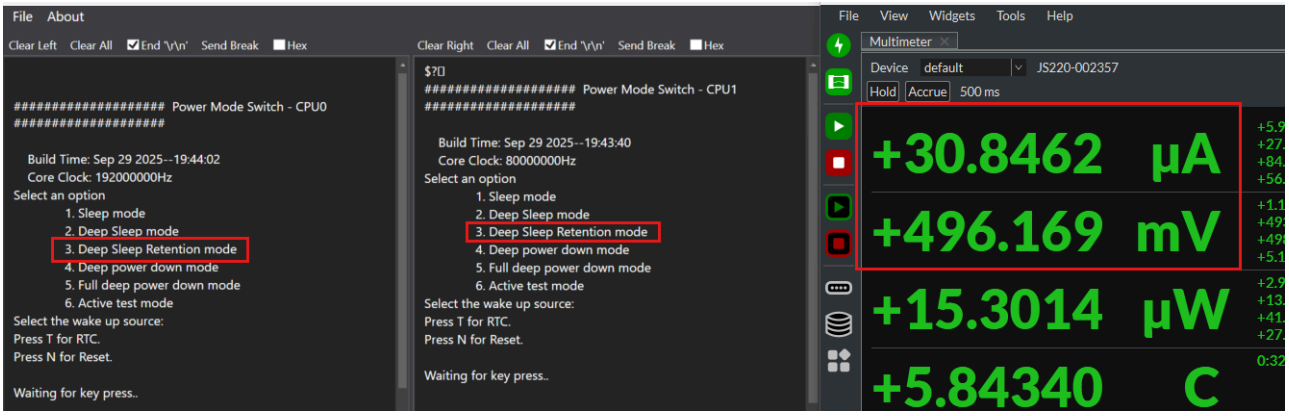

Figure 26. Deep-sleep retention mode and measure JP1 current

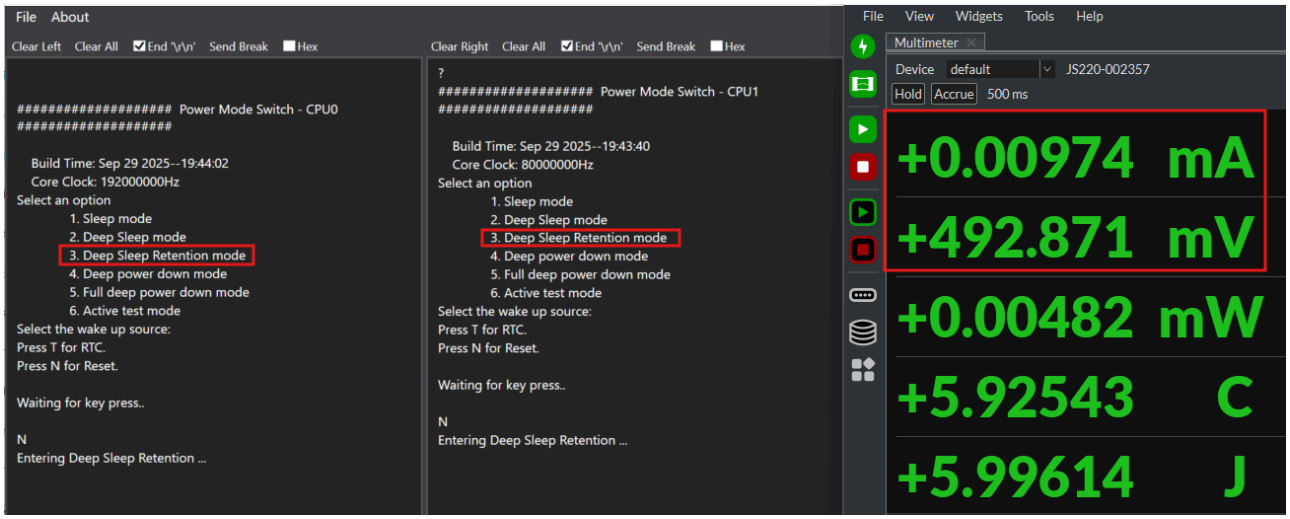

Figure 27. Deep-sleep retention mode and measure JP3 current

3 Acronyms

Table 2 lists the acronyms used in this document.

Table 2. Acronyms

Term	Description
PMIC	Power Management Integrated Circuit
RTC	Real-Time Clock
SDK	Software Development Kit

4 References

Table 3 lists the references used to supplement this document.

Table 3. Related documentation/resources

Document	Link/how to access
i.MX RT700 Crossover Microcontroller Data Sheet (document IMXRT700EC)	Contact local FAE or sales representative
MCUXpresso SDK Builder	MCUXpresso SDK Builder

5 Note about the source code in the document

Sample code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision history

Table 4 summarizes the revisions to this document.

Table 4. Revision history

Document ID	Release date	Description
AN14600 v.2.0	10 November 2025	Initial public release
AN14600 v.1.0	05 March 2025	Initial internal release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

IAR — is a trademark of IAR Systems AB.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Contents

1 Introduction 2
2 Replicate the power numbers 2
2.1 Replicate power numbers using power_mode_comp_only 4
2.1.1 Case - CPU0 CM33 CoreMark, Sense Deep Sleep 4
2.1.2 Case - CPU0 CM33 Sleep, Sense Deep Sleep 10
2.1.3 Case - Compute and Sense Deep Sleep Async 11
2.1.4 Case - CPU0 CM33 and CPU1 CM33 Deep-power-down mode 11
2.1.5 Case - CPU0 CM33 and CPU1 CM33 full Deep-power-down mode 11
2.2 Replicate power numbers using power_mode_switch 12
2.2.1 Case - CPU0 CM33 DSR, CPU1 CM33 Coremark 12
2.2.2 Case - CPU0 CM33 DSR, CPU1 CM33 Sleep 15
2.2.3 Case - CPU0 CM33 DSR, CPU1 CM33 DSR 16
3 Acronyms 17
4 References 17
5 Note about the source code in the document 17
6 Revision history 18
Legal information 19