ANALOG DEVICES

EV-COG-AD3029LZ User Guide
UG-1205

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

How to Reproduce the ADuCM3027/ADuCM3029 EEMBC Scores on the EV-COG-AD3029LZ

EVALUATION KIT CONTENTS

EV-COG-AD3029LZ
EV-GEAR-EXPANDER1Z

HARDWARE REQUIRED

EEMBC ULPBench EnergyMonitor hardware

SOFTWARE REQUIRED

EEMBC ULPBench EnergyMonitor software
IAR Embedded Workbench
Any serial monitor (PuTTY is used in this example)

GENERAL DESCRIPTION

This user guide describes how to reproduce the Embedded Microprocessor Benchmark Consortium (EEMBC®) ULPBench™ Core Profile score and the CoreMark® score for the ADuCM3027/ADuCM3029 microcontrollers.

This user guide describes the steps necessary to install the software and to set up the all of the hardware for measuring both scores.

This user guide details the energy consumed by the ADuCM3027/ADuCM3029 microcontroller in the different power modes used on the benchmark, which confirms the ADuCM3027/ADuCM3029 data sheet power specifications.

The ADuCM3027/ADuCM3029 is an ultra low power, integrated, mixed-signal microcontroller system used for processing, control, and connectivity. The microcontroller unit (MCU) subsystem is based on the Arm® Cortex®-M3 processor, a collection of digital peripherals, cache embedded SRAM and flash memory, and an analog subsystem that provides clocking, reset, and power management capabilities, along with the analog-to-digital converter (ADC).

The ADuCM3027/ADuCM3029 processor provides a collection of power modes and features, for example, dynamic and software controlled clock gating and power gating, to support extremely low dynamic power management and hibernate power management.

This user guide must be used in conjunction the ADuCM3027/ADuCM3029 data sheet when using the EV-COG-AD3029LZ evaluation board.

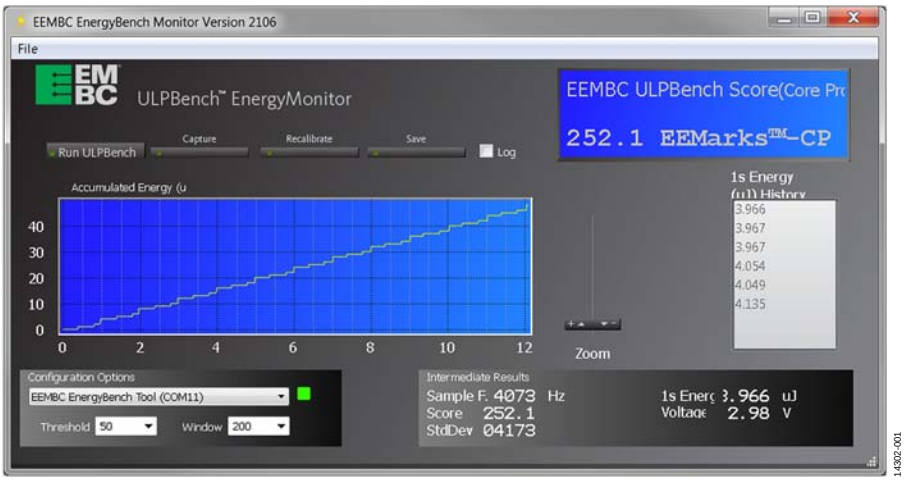

Figure 1. EEMBC ULPBench Score Window

TABLE OF CONTENTS

REVISION HISTORY

4/2018—Revision 0: Initial Version

ABOUT THE EEMBC

The EEMBC is a nonprofit industry association that detected the need for a joint democratic effort involving the leading suppliers in the embedded industry to make new benchmarks a reality.

The members of the EEMBC represent more than 40 of the leading semiconductor, intellectual property, compiler, RTOS, and system companies in the world. Furthermore, the EEMBC is licensed by more than 80 companies and more than 100 universities worldwide. Through the combined efforts of its members, EEMBC benchmarks have become an industry standard for evaluating the capabilities of embedded processors and systems according to objective, clearly defined, application-based criteria.

The EEMBC has benchmark suites targeting cloud and big data, mobiles devices (for phones and tablets), networking, ultra low power microcontrollers, the Internet of Things (IoT), digital media, automotive, and other application areas. The EEMBC also has benchmarks for general-purpose performance analysis, including CoreMark, MultiBench™ (multicore), and FPMark™ (floating point).

This user guide focuses on the CoreMark and ultra low power microcontroller benchmarks, targeted to measure the power processing and the MCU energy efficiency, respectively, because these aspects are key features of the ADuCM3027/ADuCM3029 processor.

ABOUT CoreMark

To select an MCU for a particular application, the user must know if the MCU has enough processing power to meet the requirement. Several benchmarking options are available. Dhrystone is the most widely used benchmarking option; however, it has a few inherent problems, for example, having library calls within the timed portion, and being susceptible to the ability of a compiler to optimize work. To address these problems and to provide a simple, open source benchmark, EEMBC created the CoreMark.

CoreMark is a benchmark that aims to measure the performance of central processing units (CPUs) used in embedded systems. This benchmark was developed in 2009 at the EEMBC and is intended to become an industry standard, replacing the antiquated Dhrystone benchmark. Written in C, the code contains implementations of the following algorithms:

- List processing (find and sort)
- Matrix manipulation (common matrix operations)
- State machine (determine if an input stream contains valid numbers)
- Cyclic redundancy check (CRC)

ABOUT ULPBench

Whether the target is edge nodes for the IoT or any other type of battery-powered application, the implications of ultra low power (ULP) varies. The lowest active current is required when the power source is severely limited (for example, energy harvesting). The lowest sleep current is required when the system spends most of its time in standby or sleep mode, waking up infrequently (periodically or asynchronously) to process a task. ULP can also imply great energy efficiency, where the most work is performed in a limited time period. Overall, the application requires a combination of tradeoffs on all of the previously mentioned criteria.

To ensure ULP operation over periods of months, years, and decades, application developers face a number of optimization challenges. There are an increasing number of microcontrollers claiming ULP capabilities; however, developers cannot rely on data sheet parameters alone. The EEMBC ULPBench is standardized on data sheet parameters and provides a methodology to reliably and equitably measure MCU energy efficiency.

The foundations of ULPBench are as follows:

- Comparability, to simplify the comparison of devices.
- Transparency, for measurement and the setup processes.
- Reproducibility, to simplify reproduction of the benchmark scores.

IAR SETUP

IAR TOOLS INSTALLATION

The IAR Embedded Workbench® and the included IAR C/C++ Compiler™ generates fast performing, compact code for Arm®-based applications. Analog Devices, Inc., provides the board support package for the ADuCM3027/ADuCM3029 for the IAR Embedded Workbench.

Support for the ADuCM3027/ADuCM3029 is provided within the board support package.

The Kickstart edition is a special starter kit/evaluation version of IAR that is free. This edition has limitations both in code size (32 kB) and in the service and support provided.

The IAR Embedded Workbench can be downloaded from the IAR website.

IAR PROJECT CONFIGURATION

This section describes the IAR configuration for proper operation. Only the settings that must be modified from the default values are mentioned.

1. Right-click the name of the project and click **Options…**, as shown in Figure 2.

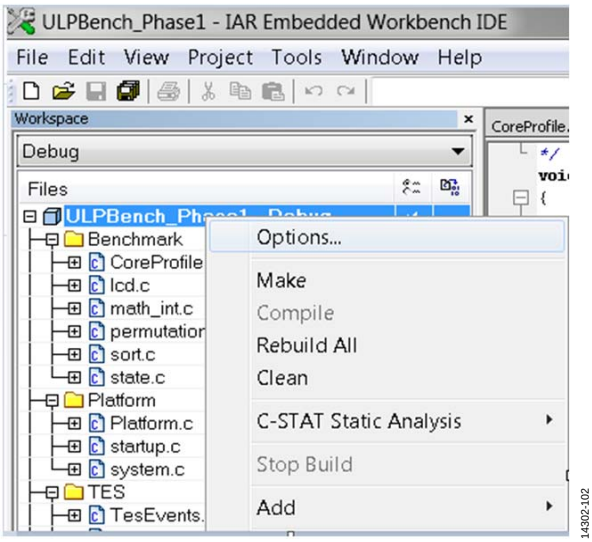

Figure 2. Project Options

2. From the **General Options** menu, in the **Target** tab, ensure that **AnalogDevices ADUCM3027** or **AnalogDevices ADUCM3029** is selected as the target in the **Processor variant** pane, depending on the microcontroller used (Figure 3).

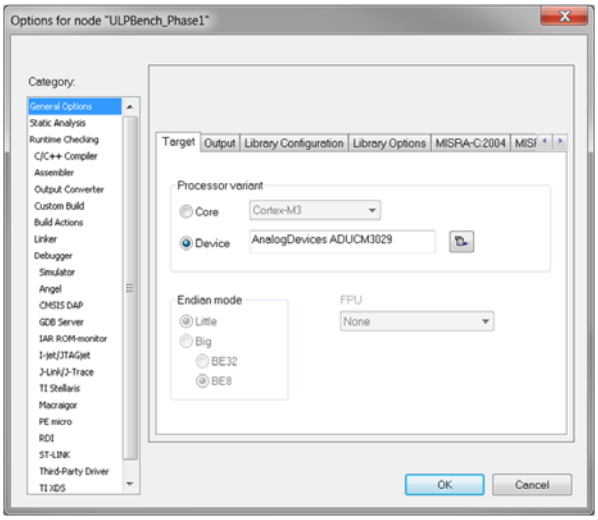

*Figure 3. **General Options**—**Target** Tab*

3. From the **C/C++ Compiler** menu, in the **Optimizations** tab, select **High** in the **Level** pane to ensure that the optimization for high speed is chosen (see Figure 4). Some functions, such as **pltInitialize**, are protected to ensure that they are not optimized. The following code protects a function and prevents the compiler from modifying the function code:

```
#pragma optimize=none
```

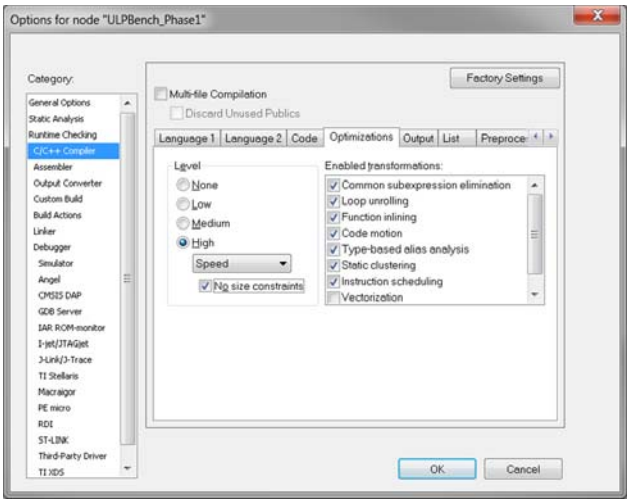

*Figure 4. **C/C++ Compiler**—**Optimizations** Tab*

Figure 5 shows the included directories path and the **Defined symbols: (one per line)** pane settings necessary for a proper compilation of the ULPBench Core Profile project.

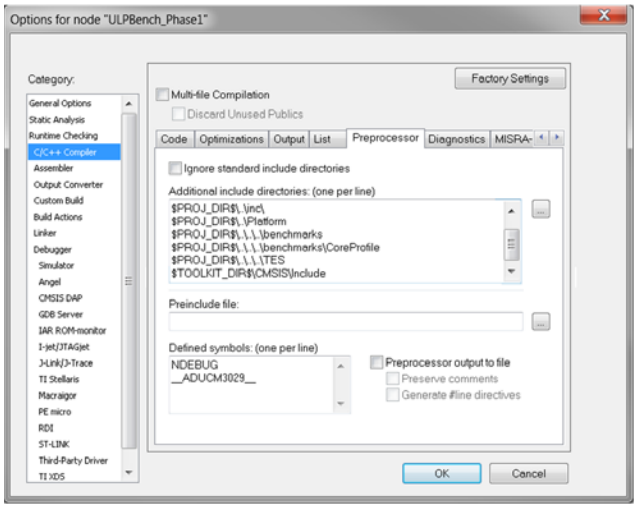

*Figure 5. **C/C++ Compiler** Menu—Preprocessor ULPBench Configuration*

Figure 6 shows the included directories path and the **Defined symbols: (one per line)** necessary for a proper compilation of the CoreMark project.

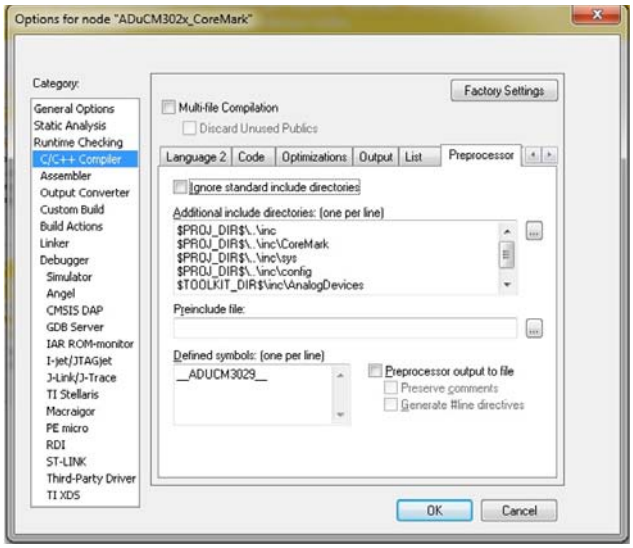

*Figure 6. **C/C++ Compiler** Menu—Preprocessor CoreMark Configuration*

To avoid undesired warnings, add the diagnostics **Pa050** and **Pa082** to the **Suppress following diagnostics** option in the **Diagnostics** tab of the **C/C++ Compiler** menu.

4. A 32-bit CRC checksum stored in the **Signature** field enables user code to request an integrity check of the user space. The user can configure the checksum as shown in Figure 7 and Figure 8 (both configurations can be found in the **Linker** menu).

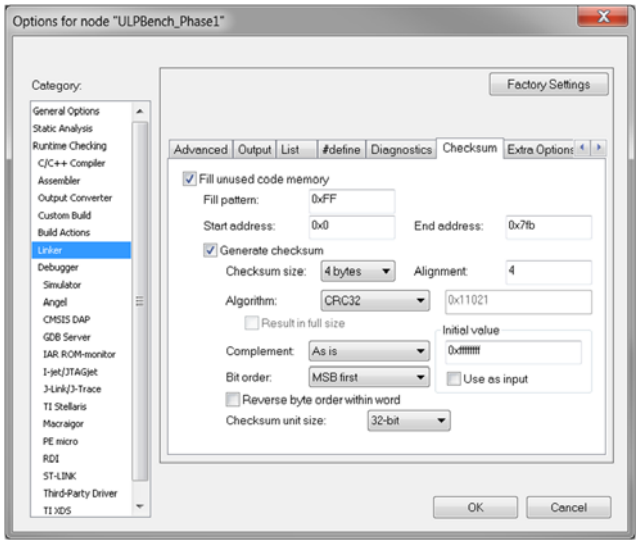

*Figure 7. **Linker** Menu—**Checksum** Tab*

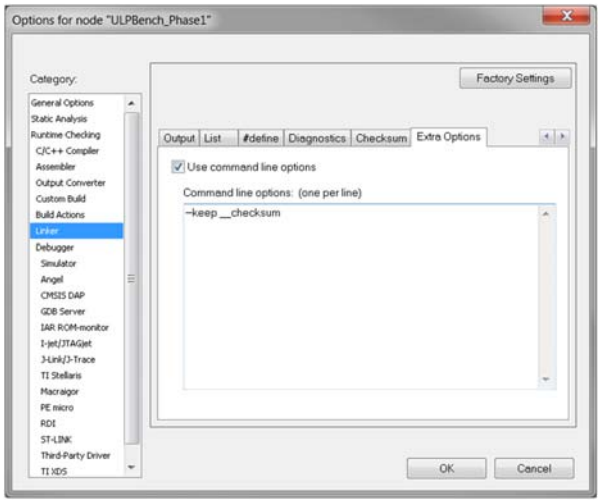

*Figure 8. **Linker** Menu—**Extra Options** Tab*

5. The debugger used is **CMSIS DAP**, selected from the dropdown list in the **Driver** pane of the **Debugger** menu (see Figure 9).

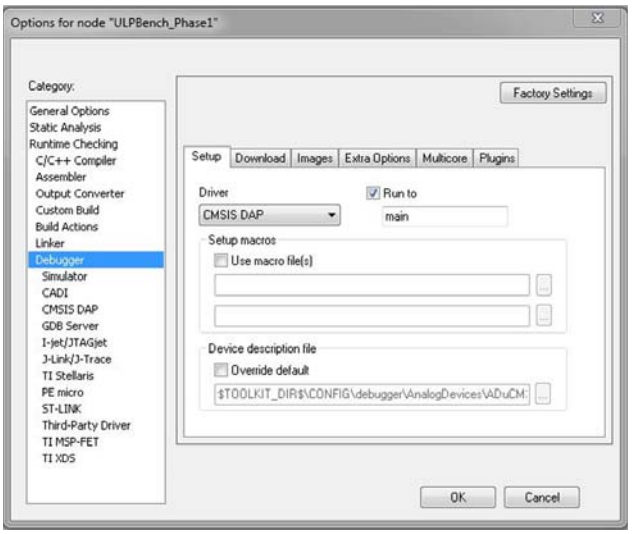

*Figure 9. **Debugger** Menu—**Setup** Tab*

6. Ensure that the **Verify download** and **Use flash loader(s)** checkboxes are selected in the **Debugger** menu in the **Download** tab, as shown in Figure 10.

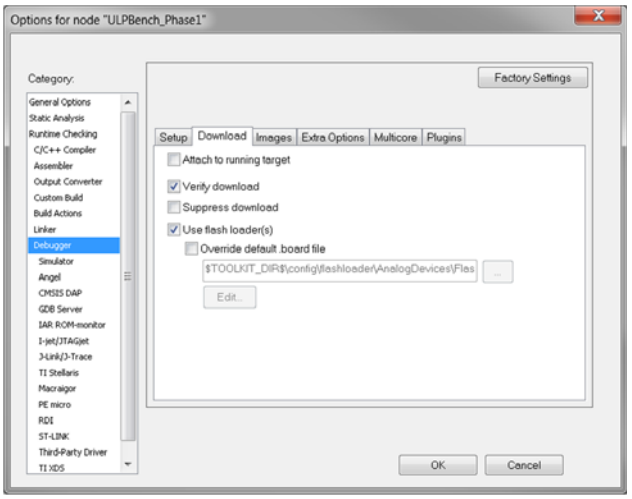

*Figure 10. **Debugger** Menu—**Download** Tab*

7. Figure 11 and Figure 12 show the **CMSIS DAP** menu configuration. From the dropdown list in the **Reset** pane, select **Hardware** for the target reset strategy.

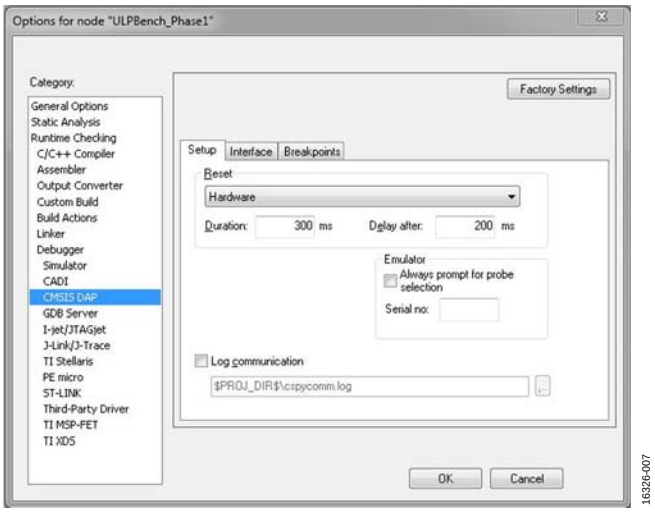

*Figure 11. **CMSIS DAP**—**Setup** Tab*

In the **Interface** tab, in the **Interface** pane, ensure that the **SWD** option is selected (see Figure 12).

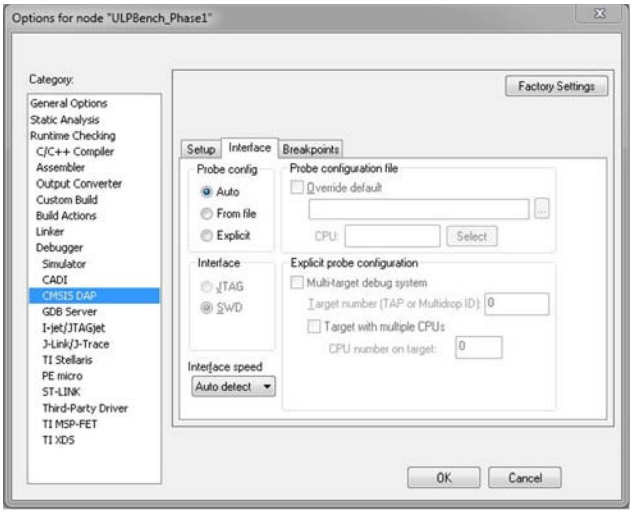

*Figure 12. **CMSIS DAP** Menu—**Interface** Tab*

CoreMark

LOADING THE CoreMark PROJECT

The project with the CoreMark source files, **core_portme.c** files, and **core_portme.h** files are tuned to the Analog Devices platform.

To add the CoreMark project on IAR, take the following steps:

1. Open the IAR Embedded Workbench.
2. Open the project in the IAR software.
3. From the **Project** menu, click **Add Existing Project…** from the dropdown list, as shown in Figure 13.

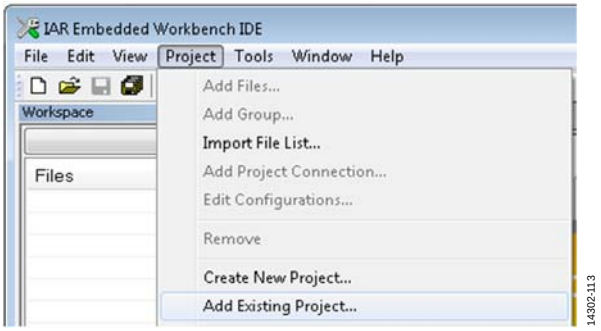

*Figure 13. **Adding Existing Project***

4. Browse through the obtained project and open the .ewp extension file. The files available in the workspace are shown in Figure 14.

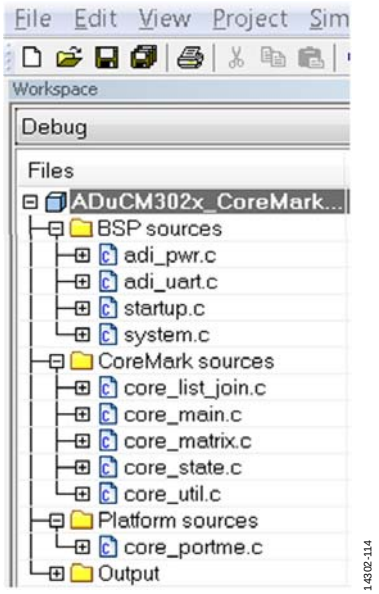

Figure 14. Project Files

The **BSP sources** folder includes the board support package files to properly configure the device. The **CoreMark sources** folder includes the source files given by the EEMBC. The **Platform sources** folder contains the **core_portme.c** file given by the EEMBC; however, the file is tuned to configure the ADuCM3027/ADuCM3029 processor.

EEMBC does not restrict on changing the **core_portme.c** and **core_portme.h** files to suit the Analog Devices platform.

The following describes attributes of the **core_portme.c** file that are not included in the file given by the EEMBC:

- Code for universal asynchronous receiver/transmitter (UART) printing.
- Code for calculating the ticks of execution using an oscillator or crystal.
- Code to configure the microcontroller properly.
- Header files to support these codes.

Device configuration. Note that the high power buck is enabled to reduce the power consumption, which is useful during power measurement when CoreMark is running (see the Power Measurements section).

The **core_portme.h** file has two defines:

- The UART_PRINT define is used to print the result through the UART. If this is commented, the results are printed only on the terminal input/output.
- The XTAL define is used to decide whether measuring the ticks using an external crystal oscillator or the internal resistor-capacitor (RC) oscillator. If this is commented, the internal oscillator is used; otherwise, the crystal oscillator is used.

RUNNING THE APPLICATION

Building the Application

To build or compile the application code, take one of the following steps:

- Click **Project**, and then click **Rebuild All** (as shown in Figure 15).

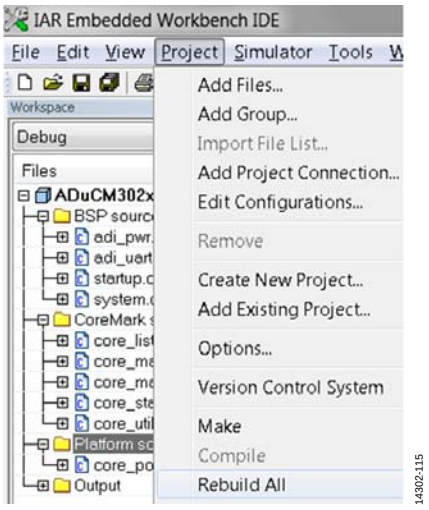

Figure 15. Build the Project

- Right-click the project name and click **Rebuild All**, as shown in Figure 16. The user is prompted to save the workspace in a .eww extension. The project must build without any errors.

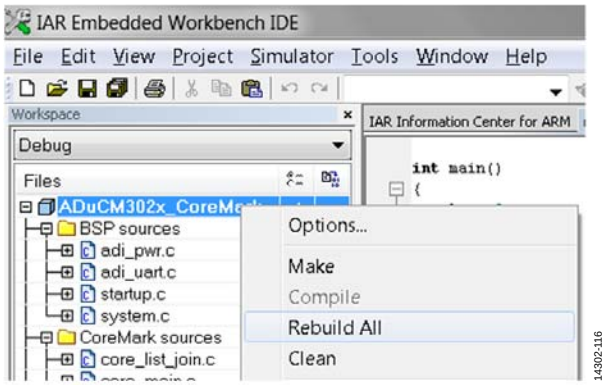

Figure 16. Building the Project

Downloading the Code

To load the code onto the EV-COG-AD3029LZ board, take one of the following steps:

- Click **Project**, **Download**, and then **Download active application**, as shown in Figure 17.
- Click **Download and Debug**, as shown in Figure 18.

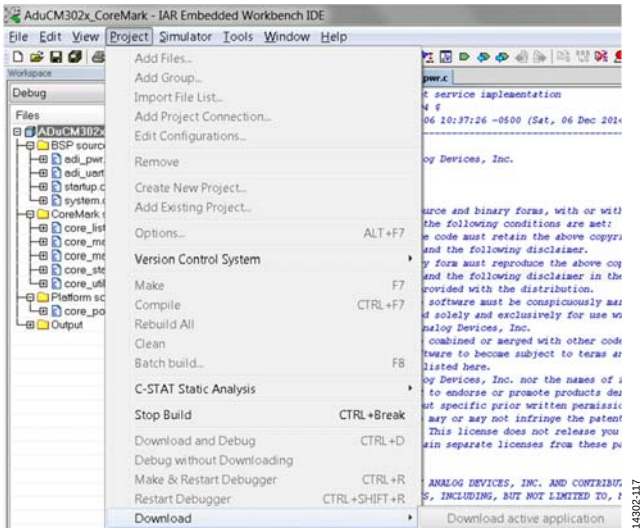

Figure 17. Downloading the Code

*Figure 18. **Download and Debug** Button*

Running the Project

To run the code, click **Go**, as shown in Figure 19.

Figure 19. Running the Project

COREMARK RESULTS

The CoreMark code must run for at least 10 seconds. The provided code has set up 10,000 iterations, which requires approximately 2 minutes to complete the execution.

The results are printed on the terminal input/output. To view the terminal input/output, click **View**, then click **Terminal I/O** from the dropdown list (see Figure 20). It is necessary to be in debug mode for this option to be active.

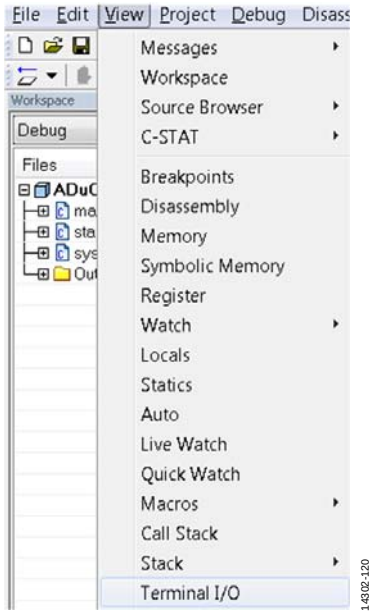

*Figure 20. Viewing the **Terminal I/O***

Figure 21. Terminal Results

By default, the UART_PRINT define in the core_portme.h file is commented.

To print the results through the UART, take the following steps:

1. Uncomment the UART_PRINT define.
2. Connect the UART port of the EV-COG-AD3029LZ to the PC using a USB cable.
3. Ensure that the UART Jumpers (1, 2, 7, and 8) of the P8 connector are connected (see Figure 22).

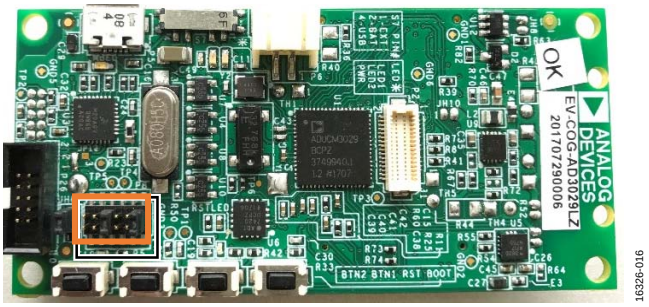

Figure 22. Jumper Connections for UART Printing

4. From the **Control Panel**, click **Device Manager**.
5. Check the COM port number to which the UART is connected.
6. Open a terminal that can connect to the UART port (the PuTTY terminal is used here).
7. Set the connection type to serial, and enter the corresponding COM port number.
8. The other settings are as shown in Figure 23.

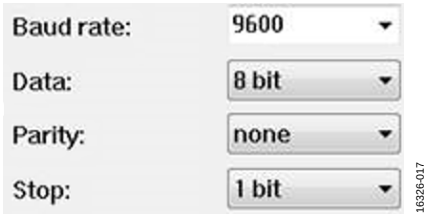

Figure 23. UART Configuration

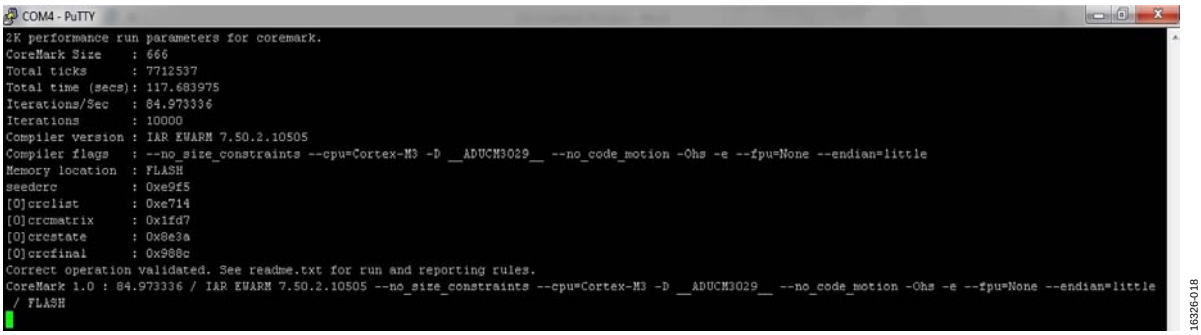

Figure 24. Results on UART

Rebuild the project and follow the instructions in the Running the Project section. The results are printed through the UART (see Figure 24).

The CoreMark number shows the raw horsepower, and the CoreMark/MHz number shows the efficiency of the core. To calculate the CoreMark/MHz number, the CoreMark number must be divided by the clock speed used when the benchmark is performed.

$$CoreMark / MHz = \frac{CoreMark\ Score}{Clock\ Frequency}$$

In this project, the ADuCM3027/ADuCM3029 processors runs at 26 MHz.

$$CoreMark/MHz = \frac{85.337589}{26\ \text{MHz}}$$

The CoreMark/MHz score is 3.2822.

This score is almost equal to the CoreMark/MHz score of the Arm® Cortex®-M3 processor, whose score is 3.32.

To report the score, CoreMark recommends the following format:

```
CoreMark/MHz 1.0: 3.2822 / IAR EWARM
7.50.2.10505 --no_size_constraints
--cpu=Cortex-M3 -D __ADUCM3029__
--no_code_motion -Ohs -e --fpu=None
--endian=little/FLASH
```

POWER MEASUREMENTS

To monitor the current consumption of the ADuCM3027/ADuCM3029 processor when it executes the CoreMark code, take the following steps:

1. Ensure that the UART_PRINT define in the **core_portme.h** file is commented so that the UART pins are not floating.
2. Load the code onto the microcontroller.
3. Remove the P8 jumpers.
4. Press the Reset button.
5. Monitor the current consumption through the TH2 jumper on the meter. The current consumption must be approximately 1165 µA when the processor executes the code at 26 MHz.
6. For dynamic current consumption, repeat this procedure with a different frequency. Change the CLKDIV definition variable to 4 to divide the frequency by 4, which yields a value of 26 MHz/4 = 6.5 MHz.
7. Monitor the current consumption on the meter. The current consumption must be approximately 425 µA.

To obtain the dynamic current consumption value, calculate the slope of the line formed by the two points (frequency and current).

The slope is calculated as follows:

$$Slope = \frac{1165 - 425}{26 - 6.5} = 38 \text{ µA/MHz}$$

The dynamic current consumption is 38 µA/MHz.

ULPBench CORE PROFILE

THE EnergyMonitor

The EEMBC ULPBench EnergyMonitor software is an accurate tool for measuring energy. The EEMBC EnergyMonitor hardware (shown in Figure 25) is needed to measure ULPBench scores. This hardware can be purchased from the EEMBC website.

Figure 25 shows the EEMBC EnergyMonitor hardware, and the VCC and GND pins used to power the EV-COG-AD3029LZ evaluation board.

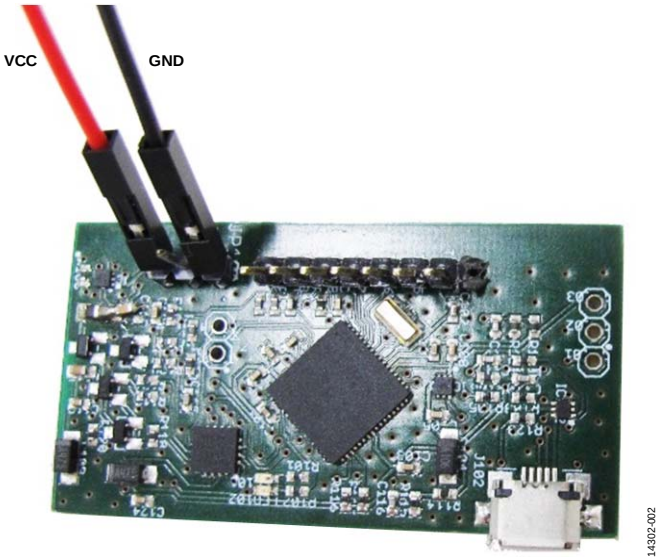

Figure 25. EEMBC EnergyMonitor Hardware

Installing the EnergyMonitor Software Drivers

When the EnergyMonitor hardware is connected to the PC for the first time, a USB driver message appears because it is an unrecognized USB device.

When the USB driver message appears, click **Next**, and then click **Manually locate USB drivers**.

If the driver message does not appear, from **Start,** open the **Device Manager** and locate the devices named **EEMBC Application UART1** and **EEMBC Energy Tool V1** to install the driver on each device.

Install the USB drivers, which are located at **/bin/USB_CDC/monitor_driver.inf** and **/bin/USB_CDC/monitor_driver.cat**. A security warning appears indicating that the publisher cannot be verified. Click **Install this driver software anyway**.

By default, 64-bit versions of Windows® Vista and later versions of Windows only load a kernel mode driver if the kernel can verify the driver signature. If using one of these versions of Windows and the drivers cannot be installed, use the appropriate mechanisms to temporarily disable the load time enforcement of a valid driver signature (the appropriate mechanism depends on the Windows version).

BUILDING THE APPLICATION CODE

To build or compile the application code, click **Project**, and then click **Rebuild All** from the dropdown list.

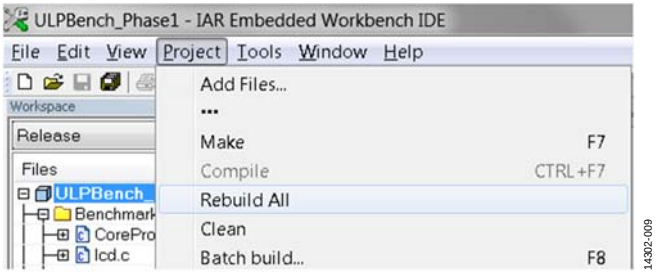

Figure 26. Build Project

Setting Up the Board to Download the Code

Download the code via USB.

If the EEMBC ULPBench_Phase1 project has been downloaded previously, the device is in hibernate mode for the majority of the time when in operation. In deep sleep mode (both hibernate and shutdown modes), the serial wire is disabled and it is not possible to download code. Press and hold the boot button, then press and release the reset button, and then release the boot button to program the device again.

The board setup for ULPBench is as shown in Figure 27. Connect the connectors from the EnergyMonitor hardware to the TH3 jumper as shown in Figure 27.

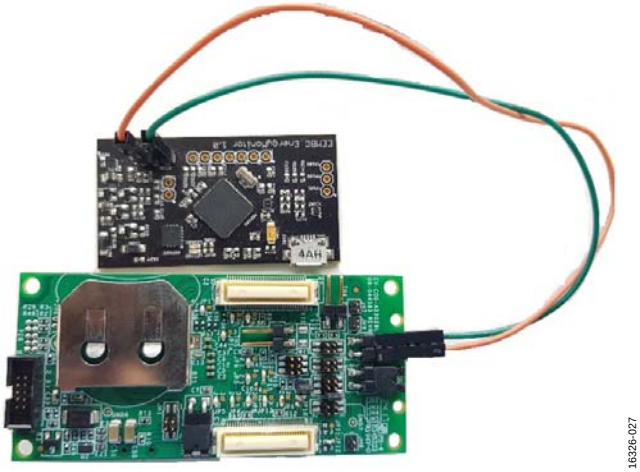

Figure 27. Hardware Setup for ULPBench

RUNNING THE BENCHMARK

Running the ULPBench Core Profile

This section provides a step by step account of how to set up the EV-COG-AD3029LZ evaluation board for measuring the ULPBench Core Profile score.

1. Connect the USB cable to the EnergyMonitor hardware.
2. Remove the debugger.
3. Remove all of the jumpers except TH2, TH1, JH4 and JH10.
4. Connect the VBAT cable from the EnergyMonitor hardware to the TH3 jumper.

Proceed to measure the score by starting the EnergyMonitor software and clicking **Run ULPBench**. The EnergyMonitor hardware powers the EV-COG-AD3029LZ evaluation board and measures the energy consumption of the core profile. At the end of the run, the software calculates the EEMBC ULPBench Core Profile score and displays it on screen. The software also displays the average energy consumed for previous cycles in the history window.

The score obtained for typical devices is around 250 EEMarks™-CP. This value can vary depending on process and temperature conditions. Figure 28 shows an example of a score for a typical device.

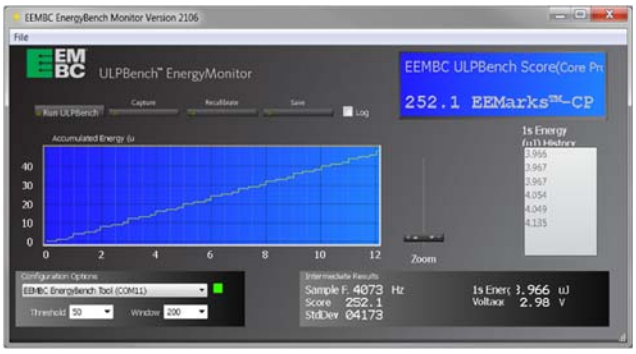

Figure 28. Example ULPBench Core Profile Score

Verifying the Proper Operation

To ensure that the workload is executing properly, a status pin (A1-1) is defined. Connect the EV-GEAR-EXPANDER1Z to the EV-COG-AD3029LZ; Pin A1-1 is configured as the status pin in the certified code.

According to the benchmark, the workload is executed twice, and at the beginning, the A1-1 status pin toggles 20×. When the second workload finishes, the A1-1 pin clears, unless an error triggers.

Figure 29 shows the output of the A1-1 pin during the execution of the workload.

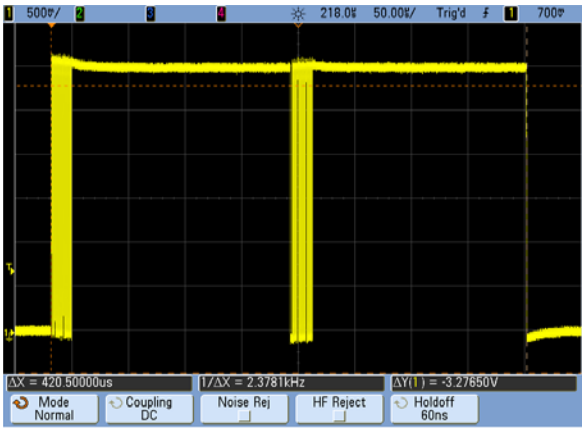

Figure 29. Verification of Proper Operation

Running the ULP Crystalless Profile

If an application does not require an accuracy as high as the one provided by a crystal, a low frequency oscillator (LFOSC) can be used as the source clock of the real time clock (RTC) to reduce the energy consumption. The LFOSC and low frequency crystal (LFXTAL) frequencies are 32 kHz.

The distributed code includes a **define** directive (Line 51 of the Platform.c file) to allow testing of the ULPBench Crystalless Profile. Uncomment the **define USE_LFOSC** line to use the LFOSC as the RTC :

```
#define USE_LFOSC
```

The score obtained for typical devices is around 265 EEMarks™-CP. This value can vary depending on process and temperature conditions. Figure 30 shows an example of a score for a typical device.

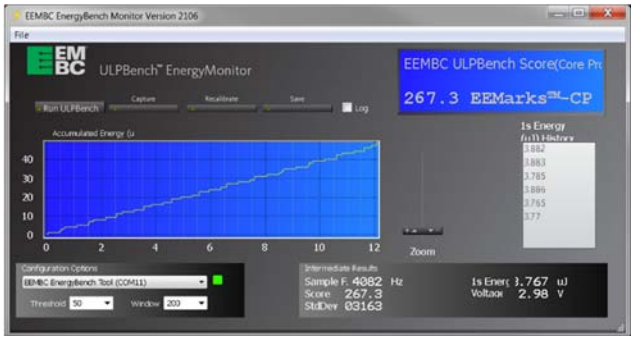

Figure 30. Example ULP Crystalless Profile Score

RESULTS ANALYSIS

The ULPMark-CP uses the following formula that takes the reciprocal of the energy values (median of 5× the average energy per second for 10 ULPBench cycles).

$$Energy\ (\mu J) = \frac{1000}{EEMarkCP}$$

The consumed energy is obtained as the sum of the energy consumed when the device is executing the workload (in active mode) and when the device is in hibernate mode.

Energy = Active Energy + Sleep Energy

Per the ADuCM3027/ADuCM3029 data sheet, the typical value for an active current is 38 µA/MHz, and 830 nA with LFXTAL and RTC enabled for a hibernate current. Figure 29 shows that the active time duration is 420 µs.

Energy = Voltage × Current × Time

Active Energy = 3 V × 1188 µA × 0.42 ms = 1.49 µJ

Sleep Energy = 3 V × 830 nA × 999.58 ms = 2.49 µJ

Per the ADuCM3027/ADuCM3029 data sheet and the execution time, the energy for the active current is 1.49 µJ, and the energy consumed during the sleep time is 2.49 µJ. The score according to those values matches the ones measured with the EEMBC EnergyMonitor software.

$$Energy\ (\mu J) = 1.49 + 2.49 = 3.98\ \mu J \cong \frac{1000}{252.1} = 3.96\ \mu J$$