

ADMT4000 Software Package Overview and Getting Started**HARDWARE REQUIRED**

- ▶ Microcontroller (any of the following)
 - ▶ [MAX32666FTHR](#)
 - ▶ [AD-APARD32690-SL](#)
 - ▶ [SDP-K1](#) controller board
 - ▶ Arduino UNO
- ▶ [EVAL-ADMT4000SD1Z](#) evaluation board
- ▶ Windows host

SOFTWARE REQUIRED

- ▶ [ADMT4000DRV-SRC Software](#) package
- ▶ No operating system (no-OS) repository

GENERAL DESCRIPTION

This user guide provides an overview of the contents of the **ADMT4000DRV-SRC Software** package for evaluating the ADMT4000.

Follow the instructions in this user guide to setup the necessary tools and dependencies to utilize the no-OS drivers and to see examples on some intended platforms.

Descriptions of the driver functions, structures, and enumerators are provided to give an overview of the coverage of the **ADMT4000DRV-SRC Software** package.

Note that the software package files were tested on a Windows[®] 10 laptop. This software package was not tested in a Linux environment, and Analog Devices, Inc., does not guarantee its use.

TABLE OF CONTENTS

Hardware Required.....	1	Installing Make.....	10
Software Required.....	1	Platform-Specific Prerequisites.....	11
General Description.....	1	Note on Analog Devices Platform.....	12
No-OS: An Overview.....	3	Building Project for Analog Devices Platform.....	13
Industrial Input and Output (IIO): An Overview.....	4	Building Project for Mbed Platform.....	14
ADMT4000DRV-SRC Software Package		ADMT4000 Driver Functions.....	15
Contents.....	5	ADMT4000 Core Functions.....	15
Admt4000 Folder.....	5	ADMT4000 Encoding (ECC and CRC)	
Platform_arduino Folder.....	5	Functions.....	15
Project Folder.....	6	ADMT4000 Supplementary Functions.....	15
Documents.....	8	ADMT4000 Driver Structures and Enums.....	17
Tools' Installation.....	9	ADMT4000 Structures.....	17
Installing Git.....	9	ADMT4000 Enums.....	17
No-OS Repository Cloning.....	9	Feedback.....	18

REVISION HISTORY**10/2024—Revision 0: Initial Version**

NO-OS: AN OVERVIEW

No-OS is the software framework created by Analog Devices to allow drivers and example projects to be compiled and used on different platforms, by using a common application programming interface (API) for common controller peripherals, such as serial peripheral interface (SPI), I²C, and general-purpose input and output (GPIO).

The **no-OS** repository is located at <https://github.com/analogdevice-sinc/no-OS>. The **ADMT4000** drivers were created to adhere to a no-OS software framework, as such, these drivers are subject to the benefits and limitations that comes with their use. Microcontrollers that can be used with the ADMT4000 under the no-OS build system include the following:

- ▶ [MAX32666FTHR](#)
- ▶ [AD-APARD32690-SL](#)
- ▶ [SDP-K1](#)

For additional information on no-OS, see the following:

- ▶ [The no-OS Overview](#)
- ▶ [The no-OS Build Guide](#)
- ▶ [The no-OS API](#)
- ▶ [The no-OS Drivers Guide](#)

INDUSTRIAL INPUT AND OUTPUT (IIO): AN OVERVIEW

Industrial input and output (IIO) is a software framework designed to provide support for typical sensors, such as analog-to-digital converters (ADCs), digital-to-analog converters (DACs), accelerometers, inertial measurement units (IMUs), and magnetometers.

Instead of accessing these devices based on their hardware protocol, such as SPI or I²C, IIO abstracts the device driver in a way that a user only sees the attributes, channels, and other properties, regardless of the kind of device.

This framework was originally designed for the Linux Kernel; however, it has a no-OS counterpart, allowing the same framework to run on microcontrollers.

For more information on the IIO framework, see these useful resources to get started:

- ▶ [The Linux driver implementer's API guide, Introduction](#) on the Linux Kernel website
- ▶ [The Linux driver implementer's API guide, Industrial I/O Devices](#) on the Linux Kernel website
- ▶ [The Linux Industrial I/O Subsystem](#)
- ▶ [no-OS IIO](#)

ADMT4000DRV-SRC SOFTWARE PACKAGE CONTENTS

Following the download and extraction of the **ADMT4000DRV-SRC Software** package from the [ADMT4000](#) page, the structure shown in [Figure 1](#) is seen.

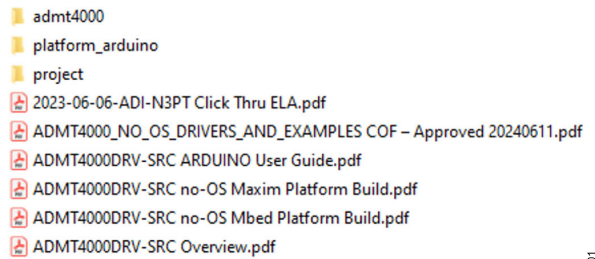


Figure 1. ADMT4000DRV-SRC Software Package Contents

ADMT4000 FOLDER

The **admt4000** folder contains the ADMT4000 and IIO ADMT4000 source and header files (see [Figure 2](#)).



Figure 2. ADMT4000 Driver Header and Source Files

The **admt4000.c** and **admt4000.h** files contain the different functions, descriptors, structures, definitions, and other necessities to set up communication between a controller and the ADMT4000. The core functions of the driver include the following:

- ▶ **admt4000_init** initializes the SPI to communicate with the ADMT4000 and the device descriptor that holds information about the device. This function also performs a reset during initialization.
- ▶ **admt4000_remove** de-allocates any resources initialized by the init function, which usually runs after the program finishes executing.
- ▶ **admt4000_read** performs a register read to retrieve the contents of one of the ADMT4000 registers.
- ▶ **admt4000_write** performs a register write to store data into one of the ADMT4000 registers.
- ▶ **admt4000_reg_update** updates specific bit or bits on a chosen ADMT4000 register using a given mask and update value.

These functions establish communication between the controller and the ADMT4000. Additional functions are available that can perform configuration of the different bit fields across the registers.

The IIO device driver contains the IIO specific implementations for the ADMT4000. The functions in this driver invoke functions from the ADMT4000 driver; however, formatted in such a way that it is compatible with IIO hosts, such as IIO oscilloscope. Items such

as, but not limited to, available measurement channels, channel attribute implementations, device-specific IIO attributes, IIO device name, and register debug functions are included in the IIO drivers. The IIO driver works together with the IIO example project to create firmware that transforms the ADMT4000 and the controller to a valid IIO device.

PLATFORM_ARDUINO FOLDER

The **ADMT4000DRV-SRC Software** package also contains some files to allow the ADMT4000 no-OS drivers to be utilized on the Arduino platform. (Note that Arduino is not a supported platform of the current no-OS architecture). The **Platform_arduino** folder contains the files shown in [Figure 3](#).

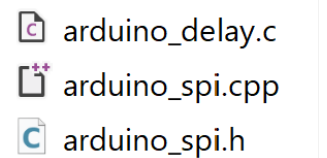


Figure 3. Arduino Platform Drivers Based on Other no-OS Platform Driver Implementations

The files in the **platform_arduino** folder, along with the ADMT4000 driver (**admt4000.c** and **admt4000.h**) and the files from the **no-OS** repository (with minor changes applied), when placed in a project folder, can help the user develop an Arduino sketch. A detailed description of the platform_arduino contents and its functions follows:

- ▶ **arduino_delay.c** is the no-OS framework compatible source file for implementing delays in Arduino.
- ▶ **arduino_spi.cpp** is the no-OS framework compatible source file for implementing SPI functions in Arduino.
- ▶ **arduino_spi.h** is the function header for the Arduino SPI platform driver containing structures and function prototypes, and so on.

For more details on setting up in Arduino environment, refer to the **ADMT4000DRV-SRC ARDUINO User Guide.docx** file that is included in the software package.

ADMT4000DRV-SRC SOFTWARE PACKAGE CONTENTS

PROJECT FOLDER

As shown in [Figure 4](#), the **project** folder contains example code to demonstrate the different functions of the [ADMT4000](#).

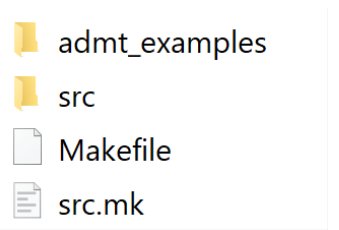


Figure 4. ADMT4000 Project Folder (no-OS Based)

As with the ADMT4000 drivers, the example projects are structured to adhere to the no-OS project structure as follows:

- ▶ The **admt_examples** folder contains the different example projects available for ADMT4000.
- ▶ The **src** folder contains the different source and header files required to run the example project on a supported platform and/or microcontroller.
- ▶ This **Makefile** is invoked to build the firmware for a target controller. It is similar to makefiles in other no-OS projects with some minor additions and changes because the drivers and other files are not yet part of the no-OS repository itself.
- ▶ The **src.mk** is a Makefile that specifies the headers and source files that must be included in the building of the project.

Example Projects

The structure of the **admt_examples** folder is shown in [Figure 5](#).

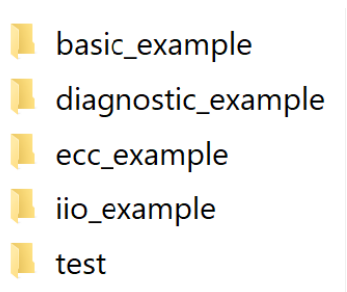


Figure 5. Available Example Projects for the ADMT4000

The example projects listed in [Figure 5](#) can be built for all supported microcontrollers as follows:

- ▶ The **basic_example** performs angle measurement repeatedly along with the turn count and the value of the fault register. The capture method can be specified as **CONTINUOUS** or **ONE-SHOT** (continuous loop).
- ▶ The **diagnostic_example** measures different diagnostic elements of the ADMT4000 such as the fault register, reference resistors state, measurement radius, fixed voltage measurement,

primary and second temperatures, and diagnostic resistor values (a continuous loop).

- ▶ The **ecc_example** showcases the error correcting code (ECC) functionality of the ADMT4000. Running this project displays the register contents and the corresponding ECC computed and actual ECC register contents for validation. Users have the option to change the register contents to display a different ECC for comparison and verification (a nonloop).
- ▶ When enabled, the **iio_example** project converts the microcontroller and the ADMT4000 to an IIO device that can then be probed with different tools, such as IIO oscilloscope and other IIO utilities (a continuous loop).
- ▶ The **test** project runs a test using the different functions implemented in the ADMT4000 drivers and returns a report that shows the number of pass and fail cases along with the places where the test failures occurred (a nonloop).

ADMT4000DRV-SRC SOFTWARE PACKAGE CONTENTS

Src Folder

As shown in [Figure 6](#), the **src** folder within the **project** folder contains the following:

- ▶ The **platform** folder contains platform-specific files, which allows the project to be built by the user on the target microcontroller.
- ▶ The **app_src.mk** makefile includes the necessary example project dependencies based on the enabled flag.

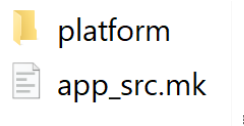


Figure 6. src Folder Contents

Platform Folder

The **platform** folder contains platform-specific files as shown in [Figure 7](#).

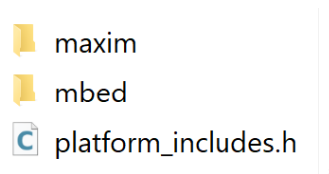


Figure 7. Platform Folder Contents

Within the **platform** folder, these subfolders indicate the name of the platform supported by the project. The **maxim** folder project supports no-OS project build using Analog Devices, Inc., microcontrollers, while the **mbed** folder has a project that is compatible with the **SDP-K1** controller board.

Refer to the [No-OS: An Overview](#) section or the [Hardware Required](#) section for a list of the supported controllers under the no-OS build system.

[Figure 8](#) shows an example of the files typically contained in the **platform** folder.



Figure 8. Platform Folder Contents

The following describes the **platform** folder contents and their functions:

- ▶ The **main (.c)** file is similar to the main file in typical embedded projects. This file displays how the enabled example project is invoked, and any other specific routines required when running

the project for a specific platform. This file also contains a check function that ensures that only one example project is enabled at any given time. Note that this file prompts an error during the build if more than one example project or no example project is enabled.

- ▶ The **parameters (.c)** file contains the different structures that are utilized by the project. These structures usually contain, but are not limited to, UART settings, SPI settings, and device-specific initial parameter structures.
- ▶ The **parameters (.h)** file stores the different macro definitions for easier identification of values across different files. The structures declared in the **parameters.c** file are also made available externally in this file with the help of an `extern` keyword, allowing the structures and other variables to be accessed elsewhere.

The **platform_includes.h** header file (as shown in [Figure 7](#)) is a consolidated header file that selects the **parameters.h** file from the available folders (for example, **maxim** or **mbed**) depending on the targeted microcontroller or platform. If the build system detects the built option is for an Analog Devices platform or microcontroller, the **platform_includes.h** file selects the **parameters.h** from the **maxim** folder.

ADMT4000DRV-SRC SOFTWARE PACKAGE CONTENTS

DOCUMENTS

In addition to the software project files, the **ADMT4000DRV-SRC Software** package also contains the following documents that help the user to get started with using the drivers and example projects and corresponding licensing related items:

- ▶ The **2023-06-06-ADI-N3PT Click Thru ELA** document is the evaluation license agreement (ELA), which is in both **.pdf** and **.rtf** file formats.
- ▶ The **ADMT4000DRV-SRC_NO_OS_DRIVERS_AND_EXAMPLES COF - Approved 20240611** document contains the general description of the ADMT4000 no-OS drivers and example projects.
- ▶ The **ADMT4000DRV-SRC ARDUINO User Guide** details how to use the driver and includes examples on the Arduino environment.
- ▶ The **ADMT4000DRV-SRC no-OS Analog Devices Platform Build** document details using the driver and examples for Analog Devices platform microcontrollers.
- ▶ The **ADMT4000DRV-SRC no-OS Mbed Platform Build** document details the driver and examples for the mbed platform ([SDP-K1](#)).
- ▶ The **ADMT4000DRV-SRC Overview** document details the general steps for getting started and definitions of some of the functions.

Before proceeding with any of the other guides, it is recommended to review the overview document first because this UG details setting up the tools and other prerequisites.

TOOLS' INSTALLATION

The tools in the [Installing Git](#), [No-OS Repository Cloning](#), and [Installing Make](#) sections are required to build a project.

INSTALLING GIT

To use the **ADMT4000DRV-SRC Software** package, it is necessary to install **Git** functionality, which enables operations such as cloning a repository to configure the software dependencies necessary to build projects for the ADMT4000.

Download and install **Git** from <https://git-scm.com/downloads>. Once installed, right-click within the Windows folder to expose the **Git** functionality, as shown in [Figure 9](#).

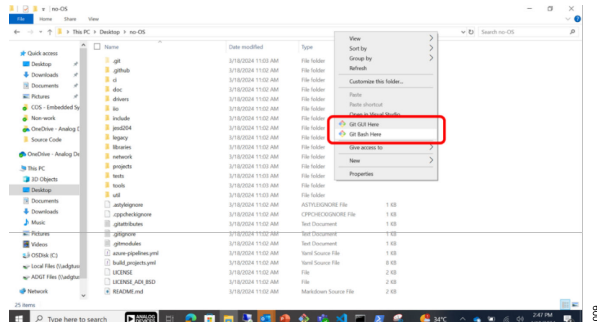


Figure 9. Git Options: Git GUI and Here and Git Bash Here

Note that, once **Git** is installed, the next step is to clone the Analog Devices no-OS repository.

NO-OS REPOSITORY CLONING

Technically, the **no-OS** repository can be cloned anywhere; however, for immediate compatibility with the files inside the **ADMT4000DRV-SRC Software** package, it is advisable to clone the repository inside the **ADMT4000DRV-SRC Software** package folder as well.

Right-click and select the **Git Bash Here** option, which opens a terminal, as shown in [Figure 10](#).

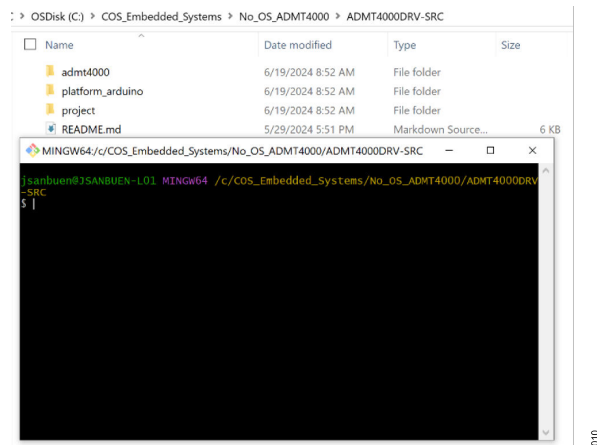


Figure 10. Git Terminal Opened with Location Inside the ADMT4000DRV-SRC Software Package Location

In the **Git** terminal, type the following:

Git clone - -recursive <https://github.com/analogdevicesinc/no-OS.git>

As a shortcut, copy the previous line and paste it into the terminal by pressing the **Shift + insert** key on your PC (contrary to the more common **Ctrl + V** shortcut). Then, observe the terminal and monitor for any screen prompts. A successful clone results in the **no-OS** folder appearing inside the **ADMT4000DRV-SRC Software** package folder (see [Figure 11](#)).

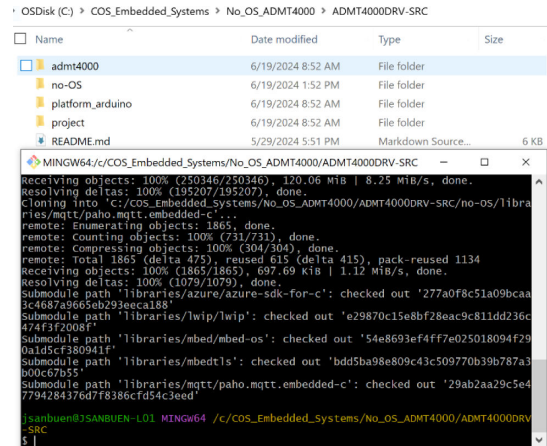


Figure 11. Successful Clone of the no-OS Repository Inside the Software Package Folder

Cloning recursively (via the **- -recursive flag**) also clones the sub-modules inside the **no-OS** main repository. Verify that the recursive clone is successful by checking the folders inside the **libraries** directory. If successful, the folders inside will not be empty. [Figure 12](#) shows an example of the **mbed-OS** folder.

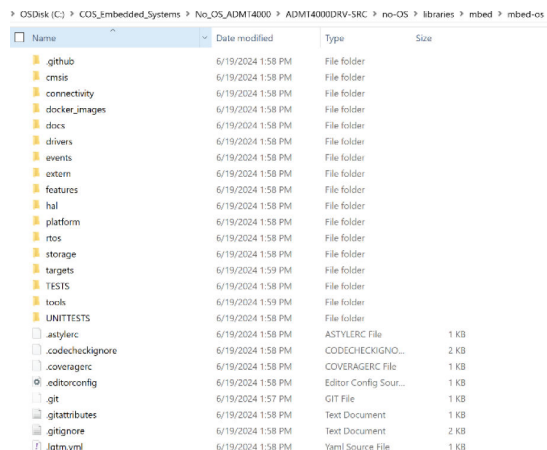


Figure 12. Nonempty mbed-OS Folder due to a Recursive Clone of the no-OS Repository

TOOLS' INSTALLATION

INSTALLING MAKE

The **Make** tool is required to build projects in the **ADMT4000DRV-SRC Software** package. The no-OS repository includes a script that can be executed in the **Git** terminal to install a compatible version of **Make**.

Open a **Git** terminal inside the **no-OS** repository by right-clicking and selecting the **Git Bash Here** as highlighted in **Figure 13**.

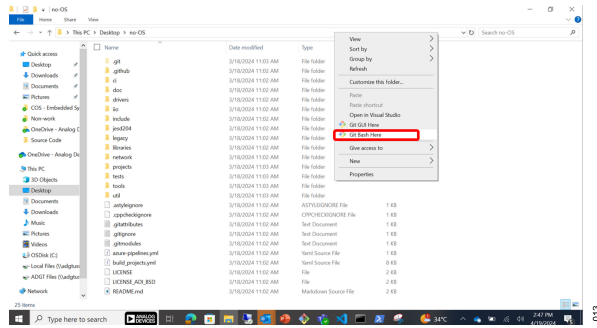


Figure 13. Git Terminal Opened from Inside no-OS Repository

Select the **Git Bash Here** option, a **Git** terminal session initiates, and a window appears inside the **no-OS** repository (see **Figure 14**).

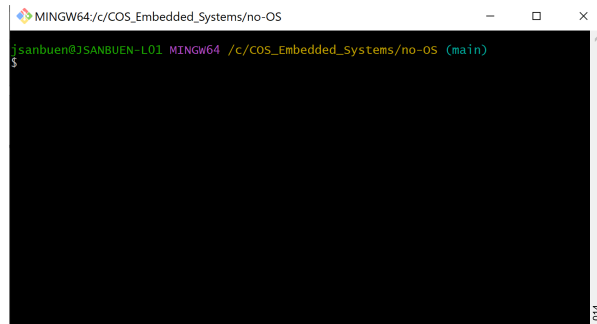


Figure 14. Git Terminal Correctly Opened from Inside the Repository

Aside from the directory showing that the user is inside the **no-OS** repository, it is also important to verify the blue text as this indicates which branch of the **no-OS** repository is available on your local copy. Users must always ensure that this blue text displays **main**.

If the blue text does not read **main**, run a **git checkout main** command to switch to the **main** branch.

Once this is completed, type the following in the **Git** terminal to run the installation script for **Make**:

```
./tools/scripts/git-bash.sh
```

To verify the success of the **Make** installation run a **make -version** to display the **Make** version installed on your PC, as shown in **Figure 15**.

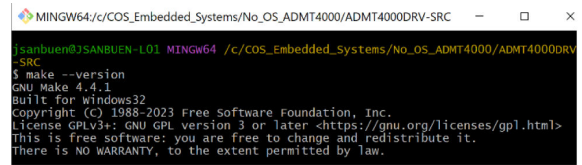


Figure 15. Make Version Running on Windows PC | Make 4.4.1

PLATFORM-SPECIFIC PREREQUISITES

Tools and dependencies for the no-OS supported target microcontroller must be installed.

For details, refer to the [No-OS Build Guide](#). Refer to the Build Prerequisites section of this guide (Windows development and testing are all done in windows units).

NOTE ON ANALOG DEVICES PLATFORM

When using the [MAX32666FTHR](#) controller with the [ADMT4000](#), an error condition may occur that results in the system not being detected as an IIO device. In this case, it is recommended to use an earlier version of the MAX32666FTHR software development kit (SDK). Earlier versions of the SDK can be found at <https://github.com/analogdevicesinc/msdk/releases>. The June 2023 release is recommended by Analog Devices.

BUILDING PROJECT FOR ANALOG DEVICES PLATFORM

When building the example projects using the no-OS build system and [MAX32666FTHR](#) microcontroller, refer to the **ADMT4000DRV-SRC no-OS Maxim Platform Build** document provided in the **ADMT4000DRV-SRC Software** package. This manual includes hardware and software instructions to help users build a project with the MAX32666FTHR microcontroller setup.

BUILDING PROJECT FOR MBED PLATFORM

For building example projects using the no-OS build system for an [SDP-K1](#), refer to the **ADMT4000DRV-SRC no-OS Mbed Platform Build** document that comes with the **ADMT4000DRV-SRC Software** package. This manual includes hardware and software instructions to help users build a project using the SDP-K1 setup.

ADMT4000 DRIVER FUNCTIONS

The following sections give a brief description of the functions available in the [ADMT4000](#) driver. For more details especially on the input arguments, refer to the [admt4000.c](#) and [admt4000.h](#) files because these files include docstring comments for documentation purposes as well.

ADMT4000 CORE FUNCTIONS

The ADMT4000 core functions allow a controller to communicate with the ADMT4000. These core functions are the bare minimum, and the foundation of the other supplementary functions as well.

Table 1. ADMT4000 Core Function

Function Name	Function Description
<code>admt4000_init()</code>	It initializes a SPI soft reset to clear the contents of fault register. The device descriptor is also initialized by invoking other functions such as <code>set_page()</code> to initialize the page that the ADMT4000 is currently at.
<code>admt4000_remove()</code>	It de-allocates any resources instantiated by the <code>admt4000_init()</code> function. Note you must run this function when the example program is done to ensure that dynamically allocated resources are freed before total exit.
<code>admt4000_write()</code>	It writes to the ADMT4000 registers, which includes the cyclic redundancy check (CRC).
<code>admt4000_read()</code>	It reads from the ADMT4000 registers, which includes the CRC.
<code>admt4000_reg_update()</code>	It updates certain bit fields in the registers by using update mask and value. This function is used by other driver functions as well.

ADMT4000 ENCODING (ECC AND CRC) FUNCTIONS

The ADMT4000 encoding functions are used for performing code checking as part of the functionality of the ADMT4000 (CRCs for read and write commands and ECCs for register content validation).

Table 2. ADMT4000 Encoding Functions

Function Name	Function Description
<code>admt4000_compute_crc()</code>	It computes CRCs for register data (writes and reads).
<code>admt4000_ecc_encode()</code>	It encodes input data streams.
<code>admt4000_hamming_calc()</code>	It computes Hamming distance given bit position, code length, and actual code.
<code>admt4000_error_check()</code>	It checks if input data has errors based on the Hamming calculation.
<code>admt4000_update_ecc()</code>	It is the specific routine of ECCs for ADMT4000. It is invoked by other functions that modify the configuration registers. It combines <code>ecc_encode()</code> and <code>hamming_calc()</code> .

ADMT4000 SUPPLEMENTARY FUNCTIONS

The ADMT4000 supplementary functions implemented here are a combination of the core functions that perform specific functions by interacting with certain bit fields across different registers.

Table 3. ADMT4000 Supplementary Functions

Function Name	Function Description
<code>admt4000_set_page()</code>	It sets the page to either 0 (true) or 2 (false).
<code>admt4000_get_page()</code>	It gets the current page setting.
<code>admt4000_set_cnv()</code>	It sets or resets the CNV bits to mimic the rising or falling edge on the CNV pin.
<code>admt4000_get_cnv()</code>	It gets the value of the CNV bits.
<code>admt4000_raw_angle_read()</code>	It reads the ABSANGLE and ANGLE registers in one coordinated system (CS) frame.
<code>admt4000_get_raw_turns_and_angles()</code>	It parses the data from the ABSANGLE and ANGLE registers.
<code>admt4000_get_converted_turns_and_angles()</code>	It converts raw and parsed data to physical measurement.
<code>admt4000_set_gpio()</code>	It sets the general-purpose input and output (GPIO) level.
<code>admt4000_get_gpio()</code>	It gets the GPIO level.
<code>admt4000_clear_faults()</code>	It clears the content of fault register.
<code>admt4000_read_fault()</code>	It reads the selected fault bit in fault register.
<code>admt4000_get_secondary_angle()</code>	It gets the secondary angle data.
<code>admt4000_get_converted_secondary_angle()</code>	It gets the converted secondary angle data.
<code>admt4000_get_angle()</code>	It obtains the angle data (SINE, COSINE, SECANLGI, and SECANGLQ).
<code>admt4000_get_radius()</code>	It gets the value of radius vector.
<code>admt4000_get_converted_radius()</code>	It gets the converted radius (mV/V).
<code>admt4000_get_ref_res_state()</code>	It gets the status of the reference resistor.
<code>admt4000_get_fixed_voltage()</code>	It gets the fixed voltage from the registers.
<code>admt4000_get_converted_fixed_voltage()</code>	It gets the converted fixed voltage (volts).
<code>admt4000_get_diag_res()</code>	It gets the positive or negative value, or both values, of the diagnostic resistor from the registers.
<code>admt4000_get_converted_diag_res()</code>	It gets the converted diagnostic resistor value (percentage).

ADMT4000 DRIVER FUNCTIONS

Table 3. ADMT4000 Supplementary Functions (Continued)

Function Name	Function Description
<code>admt4000_get_temp()</code>	It gets the primary or secondary value from the temperature registers.
<code>admt4000_get_converted_temp()</code>	It gets the converted primary and secondary temperatures (Celsius).
<code>admt4000_set_storage_config()</code>	It configures the values of the storage bits.
<code>admt4000_get_storage_config()</code>	It gets the value of the storage bits.
<code>admt4000_set_cnv_sync()</code>	It configures the start conversion sync mode.
<code>admt4000_get_cnv_sync()</code>	It obtains the start conversion sync mode value.
<code>admt4000_set_angle_filt()</code>	It enables or disables the angle filter.
<code>admt4000_get_angle_filt()</code>	It checks if the filter is enabled or disabled.
<code>admt4000_set_h8_ctrl()</code>	It configures the 8 th harmonic correction source (true = user supplied).
<code>admt4000_get_h8_ctrl()</code>	It gets the 8 th harmonic correction source configuration.
<code>admt4000_set_seq_mode()</code>	It configures the ADMT4000 sequencer mode.
<code>admt4000_get_seq_mode()</code>	It gets the configured sequencer mode.
<code>admt4000_set_cnv_mode()</code>	It sets the angle conversion mode (one-shot or continuous).
<code>admt4000_get_cnv_mode()</code>	It gets the angle conversion mode.
<code>admt4000_io_en()</code>	It enables or disables the selected GPIO.
<code>admt4000_gpio_func()</code>	It configures the selected GPIO as a regular GPIO or for an alternate function.
<code>admt4000_set_angle_thres()</code>	It configures the angle threshold register value.
<code>admt4000_set_converted_angle_thres()</code>	It configures the angle threshold (degrees).
<code>admt4000_get_angle_thres()</code>	It gets the register value of the angle threshold.
<code>admt4000_get_converted_angle_thres()</code>	It gets the angle threshold in degrees.
<code>admt4000_set_cnv_cnt()</code>	It configures the CNV_CNT bits.
<code>admt4000_get_cnv_cnt()</code>	It obtains the conversion count.

Table 3. ADMT4000 Supplementary Functions (Continued)

Function Name	Function Description
<code>admt4000_set_hmag_config()</code>	It configures the HxMAG registers (x = 1, 2, 3, or 8) using raw register data.
<code>admt4000_set_converted_hmag_config()</code>	It configures the HxMAG registers using the actual value (degrees).
<code>admt4000_get_hmag_config()</code>	It gets the value of the HxMAG (x = 1, 2, 3, or 8) registers.
<code>admt4000_get_converted_hmag_config()</code>	It gets the converted values of the HxMAG (x = 1, 2, 3, or 8) registers (magnitude and degrees).
<code>admt4000_set_hpha_config()</code>	It configures the HxPHA (x = 1, 2, 3, or 8) registers using raw register data.
<code>admt4000_set_converted_hpha_config()</code>	It configures the HxPHA (x = 1, 2, 3, or 8) registers using the actual values (degrees).
<code>admt4000_get_hpha_config()</code>	It gets the values of the HxPHA (x = 1, 2, 3, or 8) registers.
<code>admt4000_get_converted_hpha_config()</code>	It gets the converted values of the HxPHA (x = 1, 2, 3, or 8) registers (magnitude and degrees).
<code>admt4000_get_id()</code>	It gets the unique ID values from the registers (0 to 3).
<code>admt4000_ecc_config()</code>	It enables or disables ECC functionality.

ADMT4000 DRIVER STRUCTURES AND ENUMS

Short descriptions of the available structures and enums specific for the [ADMT4000](#) follow in the [ADMT4000 Structures](#) and [ADMT4000 Enums](#) sections.

ADMT4000 STRUCTURES

Table 4. ADMT4000 Structures

Structure Name	Structure Description
<code>struct admt4000_dev</code>	This device descriptor holds the device information and is accessed by functions at run time.
<code>struct admt4000_init_param</code>	It holds the initial parameters passed to <code>admt4000_init()</code> for populating the device descriptor.
<code>struct admt4000_gpio</code>	It represents the ADMT4000 GPIO, indicating the logic level and function (regular or alternate).

ADMT4000 ENUMS

Table 5. ADMT4000 Enums

Enum Name	Enum Description
<code>enum admt4000_faults</code>	This enum describes the available faults for monitoring in the fault register.
<code>enum admt4000_vdd</code>	It describes the possible VDD configurations: 3.3 V or 5 V.
<code>enum admt4000_cnv_sync</code>	It describes the available conversion synchronization options.
<code>enum admt4000_angle_type</code>	It describe the different angle types (that is, cosine and sine).
<code>enum admt4000_angle_eck_type</code>	It lists the possible angle threshold types.

FEEDBACK

For any issues not covered in this user guide, contact magnetics@analog.com for further assistance. When doing so, specify the following:

- ▶ The evaluation board number (for example, EVAL-ADMT4000SD1Z)
- ▶ A picture of the setup
- ▶ The microcontroller used
- ▶ A summary of the issue



ESD Caution

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

Legal Terms and Conditions

By using the evaluation board discussed herein (together with any tools, components documentation or support materials, the "Evaluation Board"), you are agreeing to be bound by the terms and conditions set forth below ("Agreement") unless you have purchased the Evaluation Board, in which case the Analog Devices Standard Terms and Conditions of Sale shall govern. Do not use the Evaluation Board until you have read and agreed to the Agreement. Your use of the Evaluation Board shall signify your acceptance of the Agreement. This Agreement is made by and between you ("Customer") and Analog Devices, Inc. ("ADI"), with its principal place of business at Subject to the terms and conditions of the Agreement, ADI hereby grants to Customer a free, limited, personal, temporary, non-exclusive, non-sublicensable, non-transferable license to use the Evaluation Board FOR EVALUATION PURPOSES ONLY. Customer understands and agrees that the Evaluation Board is provided for the sole and exclusive purpose referenced above, and agrees not to use the Evaluation Board for any other purpose. Furthermore, the license granted is expressly made subject to the following additional limitations: Customer shall not (i) rent, lease, display, sell, transfer, assign, sublicense, or distribute the Evaluation Board; and (ii) permit any Third Party to access the Evaluation Board. As used herein, the term "Third Party" includes any entity other than ADI, Customer, their employees, affiliates and in-house consultants. The Evaluation Board is NOT sold to Customer; all rights not expressly granted herein, including ownership of the Evaluation Board, are reserved by ADI. CONFIDENTIALITY. This Agreement and the Evaluation Board shall all be considered the confidential and proprietary information of ADI. Customer may not disclose or transfer any portion of the Evaluation Board to any other party for any reason. Upon discontinuation of use of the Evaluation Board or termination of this Agreement, Customer agrees to promptly return the Evaluation Board to ADI. ADDITIONAL RESTRICTIONS. Customer may not disassemble, decompile or reverse engineer chips on the Evaluation Board. Customer shall inform ADI of any occurred damages or any modifications or alterations it makes to the Evaluation Board, including but not limited to soldering or any other activity that affects the material content of the Evaluation Board. Modifications to the Evaluation Board must comply with applicable law, including but not limited to the RoHS Directive. TERMINATION. ADI may terminate this Agreement at any time upon giving written notice to Customer. Customer agrees to return to ADI the Evaluation Board at that time. LIMITATION OF LIABILITY. THE EVALUATION BOARD PROVIDED HEREUNDER IS PROVIDED "AS IS" AND ADI MAKES NO WARRANTIES OR REPRESENTATIONS OF ANY KIND WITH RESPECT TO IT. ADI SPECIFICALLY DISCLAIMS ANY REPRESENTATIONS, ENDORSEMENTS, GUARANTEES, OR WARRANTIES, EXPRESS OR IMPLIED, RELATED TO THE EVALUATION BOARD INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, TITLE, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IN NO EVENT WILL ADI AND ITS LICENSORS BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES RESULTING FROM CUSTOMER'S POSSESSION OR USE OF THE EVALUATION BOARD, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DELAY COSTS, LABOR COSTS OR LOSS OF GOODWILL. ADI'S TOTAL LIABILITY FROM ANY AND ALL CAUSES SHALL BE LIMITED TO THE AMOUNT OF ONE HUNDRED US DOLLARS (\$100.00). EXPORT. Customer agrees that it will not directly or indirectly export the Evaluation Board to another country, and that it will comply with all applicable United States federal laws and regulations relating to exports. GOVERNING LAW. This Agreement shall be governed by and construed in accordance with the substantive laws of the Commonwealth of Massachusetts (excluding conflict of law rules). Any legal action regarding this Agreement will be heard in the state or federal courts having jurisdiction in Suffolk County, Massachusetts, and Customer hereby submits to the personal jurisdiction and venue of such courts. The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Agreement and is expressly disclaimed.

