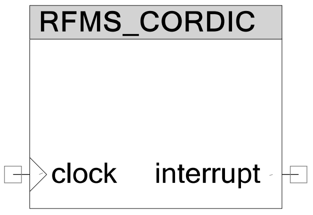
RFMS_CORDIC
2.0



Features

- Implements a 16-bit fixed point CORDIC
- Rotation mode for calculating the sine and the cosine of an angle
- Vectoring mode for calculating the arctan2 and the hypotenuse of two vectors
- PSoC 3 and PSoC 5LP support
- Software configuration only needed prior to and post calculation

General Description

The RFMS_CORDIC component implements a 16-bit fixed point CORDIC algorithm using the UDBs on PSoC 3 and PSoC 5LP devices. It supports two modes: Rotation mode and Vectoring mode. The Rotation mode calculates the sine and cosine of a given angle. The Vectoring mode accepts two vectors and calculates the angle between them (arctan2) and the hypotenuse of the vectors. The user needs only to start the component with the desired angle value (for Rotation mode) or the vectors (for Vectoring mode), and retrieve the results when the component interrupt is triggered.

Input/Output Connections

This section describes the input and output connections of RFMS_CORDIC.

clock - Input

The clock used to drive RFMS_CORDIC's UDB circuitry.

interrupt - Output

This pin is used to signal that the calculation is complete.

Component Parameters

The RFMS_CORDIC component has one parameter.

CORDIC_MODE

Chooses the mode in which the component operates.

- CORDIC_MODE = Vectoring Mode, calculates the arctan2 and the hypotenuse of 2 vectors.
- CORDIC_MODE = Rotation Mode, calculates the sine and the cosine of an angle.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function together with related constants provided by the “include” files. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name “RFMS_CORDIC_1” to the first instance of a component in a given project. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is “RFMS_CORDIC.”

Functions	Description
RFMS_CORDIC_Start()	Starts the CORDIC calculation
RFMS_CORDIC _Stop()	Stops the current CORDIC calculation
RFMS_CORDIC _Init()	Initializes the component
RFMS_CORDIC _Enable()	Enables the component
RFMS_CORDIC_GetData()	Retrieves the results

void RFMS_CORDIC_Start(uint16 desired_angle, uint16 x, uint16 y)

Description: This function initializes the RFMS_CORDIC hardware module as follows:

- Reset and return to idle state
- Reinitialize the registers and signals and pass the desired angle value
- Enable the component

Parameters: uint16 desired angle: The angle used to calculate the sine and cosine values when in Rotation mode (Not used when in Vectoring mode)

uint16 x: One of the vectors used in Vectoring mode (Not used in Rotation mode). Typically the value of the cosine component in the calculated hypotenuse.

uint16 y: One of the vectors used in Vectoring mode (Not used in Rotation mode). Typically the value of the sine component in the calculated hypotenuse.

Return Value: None

Side Effects: None

void RFMS_CORDIC_Stop(void)

Description: Disables the RFMS_CORDIC calculation.

Parameters: None

Return Value: None

Side Effects: This action pauses the CORDIC calculation. Re-enabling in the middle of calculation may not produce the correct answer.

void RFMS_CORDIC_Init(void)

Description: Performs initialization of the Angle LUT DMA and the required initial configuration.

Parameters: None

Return Value: None

Side Effects: None

void RFMS_CORDIC_Enable(void)

Description: Enables the RFMS_CORDIC module.

Parameters: None

Return Value: None

Side Effects: None

void RFMS_CORDIC_GetData(int16 *result_0, int16 *result_1)

Description: Retrieves the sine and cosine results in Rotation mode and retrieves the hypotenuse and arctan2 results when in Vectoring mode. This function should be called when the interrupt is triggered.

Parameters: int16 * result_0: Pointer to the first results registers. Reads the value of sine in Rotation mode, and reads the value of arctan2 in Vectoring mode.

int16 * result_1: Pointer to the second results registers. Reads the value of cosine in Rotation mode, and reads the value of hypotenuse in Vectoring mode.

Return Value: None

Side Effects: None

DC and AC Electrical Characteristics

N/A

Component Changes

This section lists the major changes in the component from the previous versions.

Version	Description of Changes	Reason for Changes / Impact
2.0	Vectoring mode added	Vectoring mode is the inverse procedure of Rotation mode. The modes can be chosen through the CORDIC_MODE parameter in the customizer.
	Added PSoC 5LP support	Previously supported only PSoC 3.
1.0	This is the first version of this component.	

© Cypress Semiconductor Corporation, 2010-2013. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.