

PSOC™ Control C3 – CORDIC

Customer training workshop (CTW)

February 2026

public

Target products

– Target product list for this training material:

Family category	Series	Code flash memory size
PSOC™ Control C3 Entry Line	PSC3P2xD	Up to 256 KB
PSOC™ Control C3 Entry Line	PSC3M3xD	Up to 256 KB
PSOC™ Control C3 Main Line	PSC3P5xD	Up to 256 KB
PSOC™ Control C3 Main Line	PSC3M5xD	Up to 256 KB

– CORDIC is supported in PSC3M3xD and PSC3M5xD series only

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 CORDIC overview 4

2 Architecture and modes 7

3 Registers 12

4 APIs 16

5 Code examples 24

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 **CORDIC overview** **4**

2 Architecture and modes 7

3 Registers 12

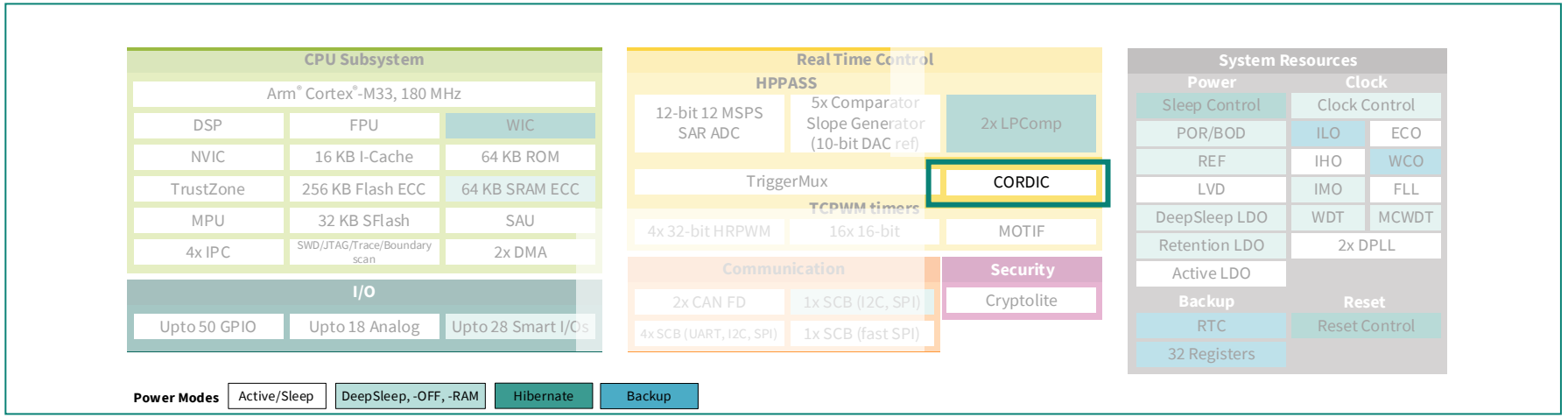
4 APIs 16

5 Code examples 24

Copyright © Infineon Technologies AG 2026. All rights reserved.

CORDIC overview

Overview



Features

- Supports CORDIC operating modes for solving circular (trigon.) and hyperbolic functions
- Vectoring and rotation modes for both functions
- Integrated look-up tables (LUTs) for all operating modes
- 24-bit input/output data width
- DMA (DW) interface signals for automatic loading of data and reading the results without host CPU intervention
- Two's complement data processing (Q format is used)
- Option to keep the last data for the next operation
- Seven clock cycles (four iterations per cycle) per calculation + 3 from a set of start (ST) bits to a set of end-of-calculation flags*.

** MMIO Write/Read clock cycles not accounted.*

Benefit

- Offload calculations from CPU and reduce processing time
- Add computational power for real-time critical tasks like
  - Field-oriented motor control
  - Power conversion, and so on
- Achieve high-frequency control loops for real-time applications

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 CORDIC overview 4

2 **Architecture and modes** **7**

3 Registers 12

4 APIs 16

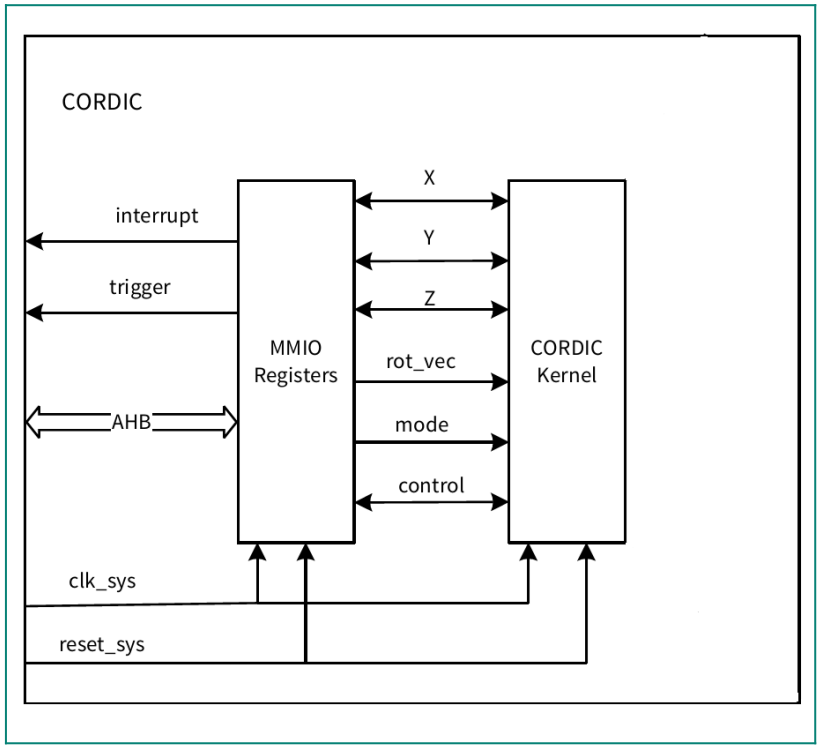
5 Code examples 24

Copyright © Infineon Technologies AG 2026. All rights reserved.

Architecture and modes

Architecture

- System Interface
  - Interrupt signal to CPU
  - Trigger to CPU/DMA for completion
  - Trigger to CPU for error
- MMIO
  - Registers for control, status and data
  - Access through the AHB slave interface
- Kernel
  - For the iterative CORDIC algorithm
  - X, Y and Z registers configured for initial data
  - Initial data registers hold intermediate data
  - Calculation using CORDIC iteration blocks
  - Iteration of the CORDIC algorithm using shifters and adders
  - Four CORDIC iteration blocks in series per clock cycle
  - Seven clock cycles for the results, which means 28 iterations
  - Final values available in X, Y, Z data registers
  - sequence is controlled by a finite state machine (FSM) and an iteration counter
  - Look up tables (LUTs) for the required constants



Hint bar

See the architecture reference manual, Architecture (section 11.2), for additional details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

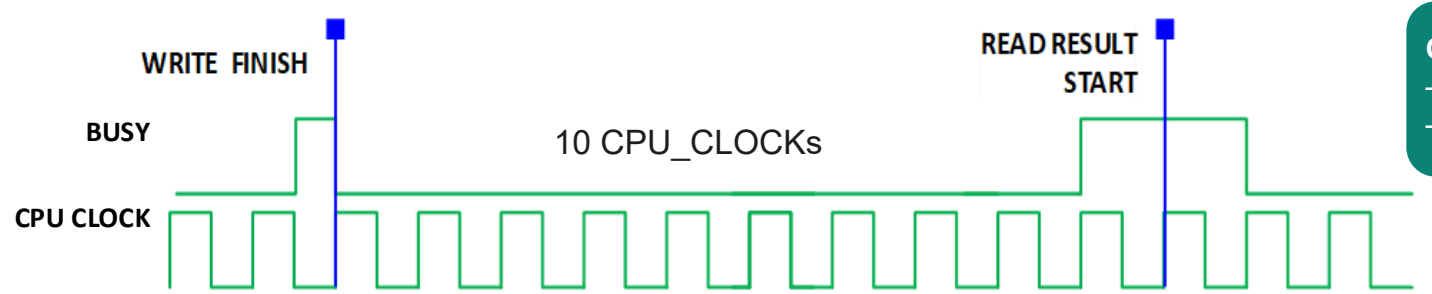
Operating modes

– General equations describing CORDIC co-processor iterative behavior:
  – $X_{I+1} = X_I - m \cdot d_I \cdot Y_I \cdot 2^{-1}$
  – $Y_{I+1} = Y_I + d_I \cdot X_I \cdot 2^{-1}$
  – $Z_{I+1} = Z_I - d_I \cdot E_I$

Function	Rotation Mode	Vectoring Mode
	$d_I = sign(z_I)$	$d_I = -sign(z_I)$
Circular	$X_{RES} = K \cdot [X_{IN}\cos(Z_{IN}) - Y_{IN}\sin(Z_{IN})]/MPS$	$X_{RES} = K \cdot sqrt(X_{IN}^2 + Y_{IN}^2)/MPS$
$m = 1$	$Y_{RES} = K \cdot [Y_{IN}\cos(Z_{IN}) + X_{IN}\sin(Z_{IN})]/MPS$	**POLAR** $Y_{RES} = 0$
$E_I = \text{atan}(2^{-1})$	**TRIGON.** $Z_{RES} = 0, K \approx 1.64676$	$Z_{RES} = Z_{IN} + \text{atan}(Y_{IN}/X_{IN})$
Hyperbolic	$X_{RES} = K \cdot [X_{IN}\cosh(Z_{IN}) + Y_{IN}\sinh(Z_{IN})]/MPS$	$X_{RES} = K \cdot sqrt(X_{IN}^2 - Y_{IN}^2)/MPS$
$m = -1$	$Y_{RES} = K \cdot [Y_{IN}\cosh(Z_{IN}) + X_{IN}\sinh(Z_{IN})]/MPS$	$Y_{RES} = 0$
$E_I = \text{atanh}(2^{-1})$	$Z_{RES} = 0, K \approx 0.82815936$	$Z_{RES} = Z_{IN} + \tanh^{-1}(Y_{IN}/X_{IN})$

SIN, COS example

Input
– $X_{IN} = 1/K$
– $Y_{IN} = 0$
– $Z_{IN} = \vartheta \rightarrow \vartheta \in [-\pi, +\pi)$



Output
– $X_{RES} = K \cdot X_{IN}\cos(Z_{IN})$
– $Y_{RES} = K \cdot X_{IN}\sin(Z_{IN})$

** MMIO Write/Read clk cycles not accounted*
10 clocks = 7 clks per calculation + 3 clocks cycles for set of start bits to set of end of calculation flags

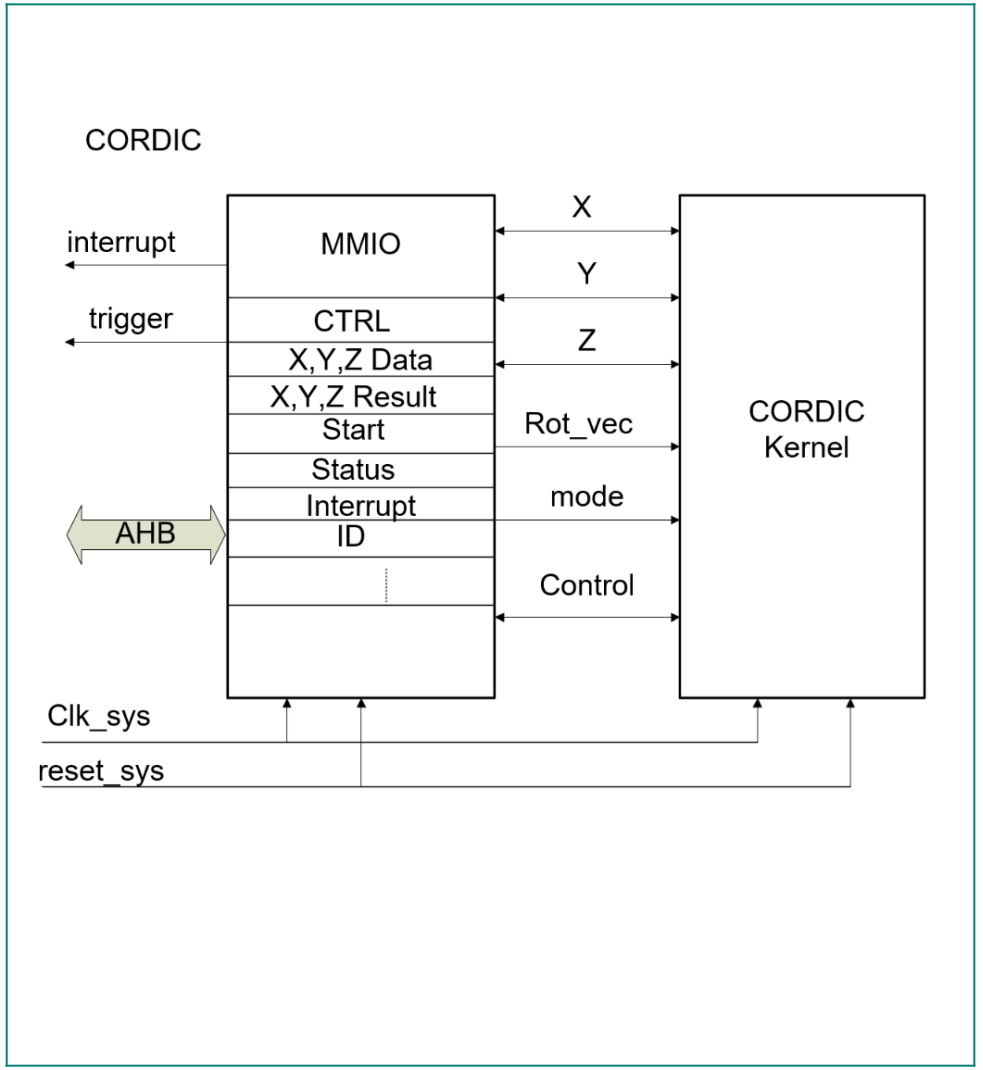
Hint bar

See the architecture reference manual, Operating modes, Chapter 11.3.3, for additional details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

Access modes

- The CORDIC block is an AHB peripheral with a standard platform-compliant interface.
- It can be accessed in the following modes:
  - **Bus hold mode**: IP holds the bus when the result register is read, when the operation is ongoing. It can read the result as soon as possible after the operation completes
  - **Polling mode**: CPU polls the Ready signal to check for completion. This allows the CPU to respond to higher-priority interrupts.
  - **DMA trigger mode**: Generate a Trigger request for a DMA transfer on completion of the operation, allowing DMA transfers from the CORDIC block.
  - **Interrupts**: Generate the interrupt when the calculation is complete



Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 CORDIC overview 4

2 Architecture and modes 7

3 **Registers** **12**

4 APIs 16

5 Code examples 24

Copyright © Infineon Technologies AG 2026. All rights reserved.

Registers

Register list

Registers	Offset	Width	Descriptions
CTL	0x0	32	Module enable/disable register
ID	0x8	32	Module identification register, set the module revision, type and number
INTR	0x10	32	Calculation complete and error interrupts cause register
INTR_SET	0x14	32	Calculation complete and error interrupts set register
INTR_MASK	0x18	32	Calculation complete and error interrupts mask register
INTR_MASKED	0x1c	32	Calculation complete and error interrupts masked register
KEEP	0x40	32	Register to select last X/Y/Z result as Initial data for new calculation
CON	0x44	32	Set Operating Mode, Rotation Vectoring Selection, Start Method, X and Y magnitude Prescaler (MPS), Number of CORDIC iterations divided by 4
CORDX	0x48	32	For initial X parameter data
CORDY	0x4c	32	For initial Y parameter data
CORDZ	0x50	32	For initial Z parameter data
CORRX	0x54	32	For X calculation result
CORRY	0x58	32	For Y calculation result
CORRZ	0x5c	32	For Z calculation result
STAT	0x60	32	Status register. Busy indication – BSY
START_CMD	0x64	32	Start command register. To start calculation, set – ST

Hint bar

See the registers reference manual, Co-Ordinate Rotation Digital Computer (CORDIC) registers, chapter 18, for more details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

Register details

- CON
  - Bit 2 and 1, set functions (circular and hyperbolic)
  - Bit 3, select rotation/vectoring mode
  - Bit 4, set start method (AUTO and ST)
  - Bit 5, result data format for X in Circular Vectoring mode
  - Bits 7 and 8, X and Y Magnitude Prescaler – MPS
  - Bit 10 to 8, number of CORDIC iterations
- CTL
  - Bit 31, enables/disables/disables the IP
- CORDX, CORDY, CORDZ
  - Initial X data, bits 31 to 8
- Keep
  - Bits 7, 6 and 5 to retain last data in X, Y and Z registers
- CORRX, CORRY, CORRZ
  - Result X, bits 31 to 8
- START_CMD
  - Bit 0, start calculation
- STAT
  - Bit 0, Busy indication – BSY

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 CORDIC overview 4

2 Architecture and modes 7

3 Registers 12

4 **APIs** **16**

5 Code examples 24

Copyright © Infineon Technologies AG 2026. All rights reserved.

APIs

PDL API types

API types	Functions description
Status	Checks if there is any running calculation
Configuration	– Clear the CORDIC Z Data register – Configures the operating modes
Non-Blocking mode	– Compute the Cosine, Sine, Tangent, for an angle in radians – Compute the principal value Arc Tangent of a given value – Compute the Hyperbolic Cosine, Sine, Tangent, Arc Tangent for an angle in radians – Get the result of a Cosine, Sine, Tangent, Arc Tangent operation – Get the result of a Hyperbolic Cosine, Sine, Tangent, Arc Tangent operation – Set the Park Transform calculation and get the Park Transform result
Blocking mode	– Compute the Cosine, Sine, Tangent, for an angle in radians – Compute the principal value Arc Tangent of a given value – Compute the Hyperbolic Cosine, Sine, Tangent, Arc Tangent for an angle in radians – Compute square root of a number
Interrupts	– Read and clear active interrupts – Set the interrupt sources – Read and set the interrupt mask – Read an active masked interrupt – Register a callback function for interrupt events

Hint bar

See the CORDIC section of the PDL/HAL API reference guide for more details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

PDL APIs

- Configuration, status and interrupt
  - `Cy_CORDIC_Enable ()`
  - `Cy_CORDIC_Disable ()`
  - `Cy_CORDIC_IsBusy ()`
  - `Cy_CORDIC_ClearZ ()`
  - `Cy_CORDIC_SetMode ()`
  - `Cy_CORDIC_GetInterruptStatus ()`
  - `Cy_CORDIC_ClearInterrupt ()`
  - `Cy_CORDIC_SetInterrupt ()`
  - `Cy_CORDIC_GetInterruptMask ()`
  - `Cy_CORDIC_SetInterruptMask ()`
  - `Cy_CORDIC_GetInterruptStatusMasked ()`
  - `Cy_CORDIC_Interrupt ()`
  - `Cy_CORDIC_RegisterCallback ()`

Hint bar

See the CORDIC section of the PDL/HAL API reference guide for more details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

PDL APIs (contd.)

- Blocking APIs
  - Cy_CORDIC_Cos ()
  - Cy_CORDIC_Sin ()
  - Cy_CORDIC_Tan ()
  - Cy_CORDIC_ArcTan ()
  - Cy_CORDIC_Cosh ()
  - Cy_CORDIC_Sinh ()
  - Cy_CORDIC_Tanh ()
  - Cy_CORDIC_ArcTanh ()
  - Cy_CORDIC_Sqrt ()
- API Implementation: Cy_CORDIC_Cos()

```
cy_en_cordic_status_t Cy_CORDIC_Cos(MXCORDIC_Type *base, CY_CORDIC_Q31_t angle_in_radians, CY_CORDIC_Q31_t *result)
{
    /* Input parameters verification */
    CY_ASSERT_L1(CY_IS_BASE_VALID(base));
    CY_ASSERT_L2(CY_CORDIC_IS_CIRCULAR_ROTATION_ANGLE_VALID(angle_in_radians));

    MXCORDIC_CON(base) = _VAL2FLD(MXCORDIC_CON_MODE, CY_CORDIC_OPERATING_MODE_CIRCULAR) |
                         _VAL2FLD(MXCORDIC_CON_ROTVEC, CY_CORDIC_ROTVEC_MODE_ROTATION)  |
                         _VAL2FLD(MXCORDIC_CON_N_ITER, CY_CORDIC_NUMBER_OF_ITERATIONS);
    MXCORDIC_CORDX(base) = _VAL2FLD(MXCORDIC_CORDX_DATA, CY_CORDIC_CIRCULAR_GAIN);
    MXCORDIC_CORDY(base) = 0UL;
    MXCORDIC_CORDZ(base) = (uint32_t)angle_in_radians;

    *result = ((CY_CORDIC_Q31_t)MXCORDIC_CORRX(base));

    return CY_CORDIC_SUCCESS;
}
```

- Non-blocking APIs
  - Cy_CORDIC_CosNB ()
  - Cy_CORDIC_SinNB ()
  - Cy_CORDIC_TanNB ()
  - Cy_CORDIC_ArcTanNB ()
  - Cy_CORDIC_CoshNB ()
  - Cy_CORDIC_SinhNB ()
  - Cy_CORDIC_TanhNB ()
  - Cy_CORDIC_ArcTanhNB ()
  - Cy_CORDIC_ParkTransformNB
  - Cy_CORDIC_GetCosResult ()
  - Cy_CORDIC_GetSinResult ()
  - Cy_CORDIC_GetTanResult ()
  - Cy_CORDIC_GetArcTanResult ()
  - Cy_CORDIC_GetCoshResult ()
  - Cy_CORDIC_GetSinhResult ()
  - Cy_CORDIC_GetTanhResult ()
  - Cy_CORDIC_GetArcTanhResult ()
  - Cy_CORDIC_GetParkResult()

Copyright © Infineon Technologies AG 2026. All rights reserved.

HAL APIs

- Configuration, status and interrupt
  - `cyhal_cordic_init ()`
  - `cyhal_cordic_free ()`
  - `cyhal_cordic_is_busy ()`
  - `cyhal_cordic_enable_output ()`
  - `cyhal_cordic_disable_output ()`
  - `cyhal_cordic_set_mode ()`
  - `cyhal_cordic_enable_event ()`
  - `cyhal_cordic_register_callback ()`

Hint bar

See the CORDIC section of the PDL/HAL API reference guide for more details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

Core steps

The following steps are to be performed using the CORDIC peripheral:

- Enable CORDIC: PDL function `Cy_CORDIC_Enable()` will enable CORDIC
- Configure the operating mode, start trigger type, data format, pre-scalar, and iteration count
- Set data and trigger the operation
- Get the result (blocking mode returns results; non-blocking mode requires polling or callback)

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 CORDIC overview 4

2 Architecture and modes 7

3 Registers 12

4 APIs 16

5 **Code examples** **24**

Copyright © Infineon Technologies AG 2026. All rights reserved.

Code examples

Code examples

- MTB App named PDL CORDIC provides a CORDIC module for quick evaluation on the PSOC™ Control C3M5 platform
- All basic CORDIC operations provided by the peripheral are demonstrated in the included code example

How it works:

- On startup, the code prints a debug menu to the UART terminal. Navigate and evaluate the CORDIC functions directly from this menu.
- The example also includes a software reference implementation (using a standard software library) for comparison

What you will see:

- The Console/Debug UART prints a menu and prompts for input
- The CE (Code Example) executes CORDIC operations using the hardware peripheral
- The CE also runs the same operations in software to provide a performance and accuracy comparison
- Results and timing for both hardware and software implementations are displayed

Hint bar

See the code example readme file for more details.

Copyright © Infineon Technologies AG 2026. All rights reserved.

Revision history

Document revision	Date	Description of change
**	2026-02-05	Initial release.

Copyright © Infineon Technologies AG 2026. All rights reserved.


Infineon