

# CLOCK\_XMC1

Home

## Apps

Here is a list of all modules:

- License Terms and Copyright Information
- Abbreviations and Definitions
- Overview
- Architecture Description
- APP Configuration Parameters
- Enumerations
- Data structures
- Methods
- Usage
- Release History

# CLOCK\_XMC1

Home

## License Terms and Copyright Information

### License Terms and Copyright Information

Copyright (c) 2015, Infineon Technologies AG All rights reserved.

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

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 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 provided with the distribution. Neither the name of the copyright holders 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.

To improve the quality of the software, users are encouraged to share modifications, enhancements or bug fixes with Infineon Technologies AG ([dave@infineon.com](mailto:dave@infineon.com)).

---

# CLOCK\_XMC1

Home

## Abbreviations and Definitions

### Abbreviations and Definitions

<b>Abbreviations:</b>	
DAVE™	Digital Application Virtual Engineer
APP	DAVE™ Application
API	Application Programming Interface
GUI	Graphical User Interface
MCU	Microcontroller Unit
SW	Software
HW	Hardware
LLD	Low Level Driver
IO	Input Output
SCU	System Control Unit
MCLK	Main Clock
PCLK	Peripheral Clock
DCO	Digitally Controlled Oscillator
OSC_HP	High Precision Oscillator
OSC_LP	Low Precision Oscillator
IDIV	Integer Divider
FDIV	Fractional Divider
MHz	Megahertz
kHz	Kilohertz

**Definitions:**

Singleton	Only single instance of the APP is permitted
Sharable	Resource sharing with other APPs is permitted
initProvider	Provides the initialization routine
Physical connectivity	Hardware inter/intra peripheral (constant) signal connection
Conditional connectivity	Constrained hardware inter/intra peripheral signal connection
Aggregation	Indicates consumption of low level (dependent) DAVE APPs



# CLOCK\_XMC1

Home

## Overview

### Overview

The **CLOCK\_XMC1 APP** provides the following functionalities:

1. Setting the sources of various clocks in xmc1 devices.
2. Configuration of Main clock.
3. Selection of fast peripheral clock.
4. Selection of RTC source
5. For XMC14 device, supported automatic DCO1 calibration based on external reference clock (OSC\_HP/OSC\_LP) during runtime to achieve a better accuracy.
6. Runtime clock changes are supported.

Main clock (MCLK) is calculated using following formula:

For XMC11/XMC12/XMC13 Device:

$$\text{MCLK} = \text{dco\_clk} / (2 * (\text{IDIV} + \text{FDIV}/256)) \text{ for IDIV}>0$$

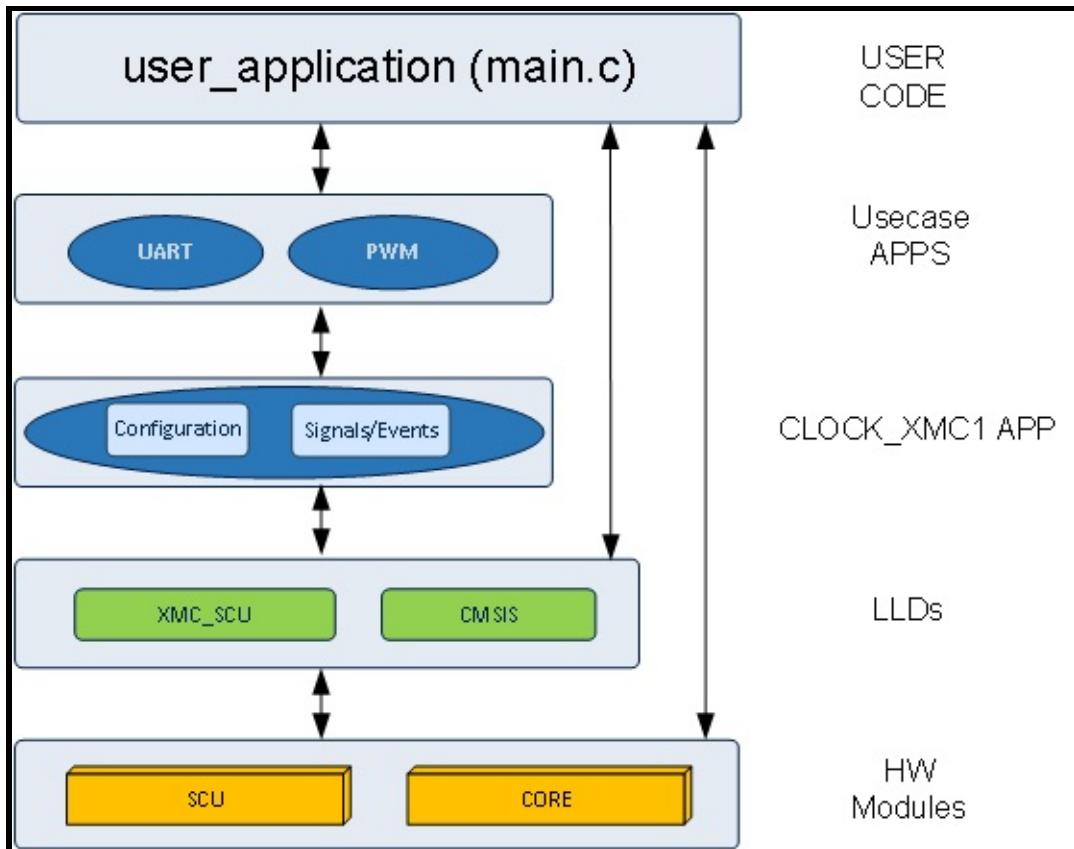
Here dco\_clk is output of the digitally controlled oscillator(DCO) which is equal to 64MHz and MCLK is required frequency.  
where *IDIV* is 8-bit integer divider and *FDIV* is 8-bit fractional divider.

For XMC14 Device:

$$\text{MCLK} = \text{DCLK} / (2 * (\text{IDIV} + \text{FDIV}/1024)) \text{ for IDIV}>0$$

Here DCLK is output of the doubler clock which is equal to twice of DCLK clock source frequency and MCLK is required frequency.  
where *IDIV* is 8-bit integer divider and *FDIV* is 10-bit fractional divider.

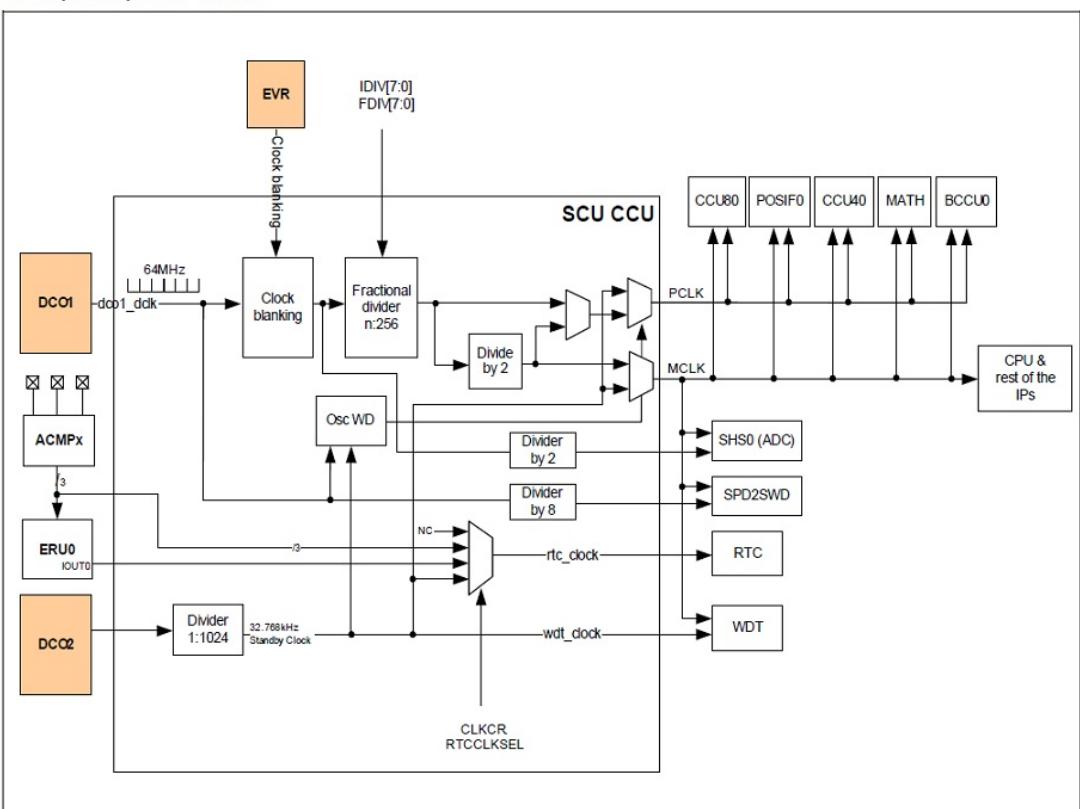
**Note:** By default the SystemInit() calls the weak API SystemCoreClockSetup(), which is presented in system\_XMC1x.c file. When the **CLOCK\_XMC1** is used in the project, then SystemInit() calls the SystemCoreClockSetup(), which is presented in **CLOCK\_XMC1** APP. This APP do not provides data structure, enums and APIs.



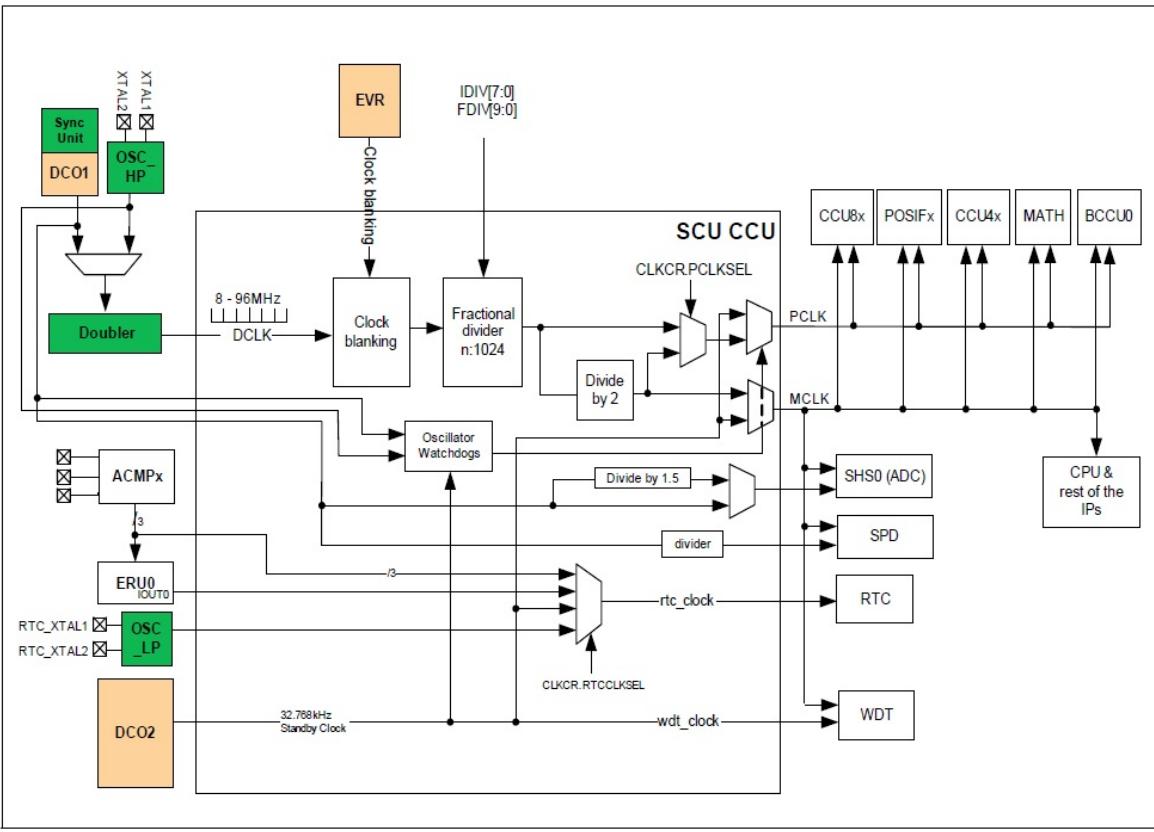
**Figure 1 :** Hardware and Software connectivity of **CLOCK\_XMC1** APP

Figure 1, shows how the APP is structured in DAVE™. The **CLOCK\_XMC1** APP uses SCU module to generate a various clocks such as: MCLK, PCLK, Standby clock etc.

XMC11/XMC12/XMC13 Device



XMC14 Device



---

## **Figure 2 : Clock selection & generation unit**

Figure 2, shows how the various clocks are derived from the source.  
*The various clocks derived from various sources:*

1. Peripherals such as CCU80, CCU40, POSIF0, MATH and BCCU0 are derived from PCLK domain.
2. The rest of the peripherals except RTC and WDT are derived from MCLK.
3. MCLK is source of core and bus system.
4. RTC and WDT are running at a frequency of 32.768 kHz from a standby clock.

## **Supported Devices**

The APP supports below devices:

1. XMC1400 Series
2. XMC1300 Series
3. XMC1200 Series
4. XMC1100 Series

## **References**

1. XMC1400 Reference Manual
2. XMC1300 Reference Manual
3. XMC1200 Reference Manual
4. XMC1100 Reference Manual

## **Limitations**

None

---

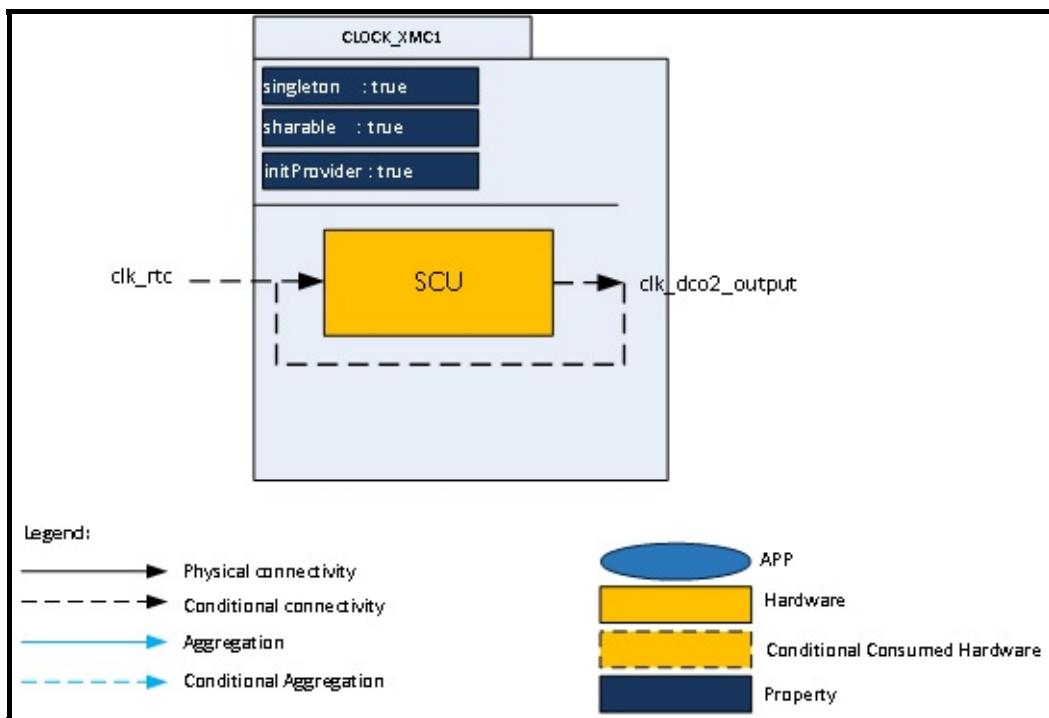
# CLOCK\_XMC1

Home

## Architecture Description

### Architecture Description

Figure 1 explains the architecture of the APP:



**Figure 1 :** Architecture of **CLOCK\_XMC1** APP

The above diagram represents the internal software architecture of the **CLOCK\_XMC1** APP. A **CLOCK\_XMC1** APP instance exists in a DAVE™ project with fixed attributes as shown. Each instance of this APP configures SCU clock module. This APP shall be used by all use-case APPs.

An instantiated APP (after code generation) generates a specific data structure with the GUI configuration.

## Signals:

The following table presents the signals provided by the APP for connection. It also gives the flexibility to configure and extend the connectivity to other APPs.

**Table 1:** APP IO signals

Signal Name	Input/Output	Availability	Description
clk_RTC	Input	Conditionally	RTC clock input selection signal. Can be connected with ERU or ANCMP APPs

# CLOCK\_XMC1

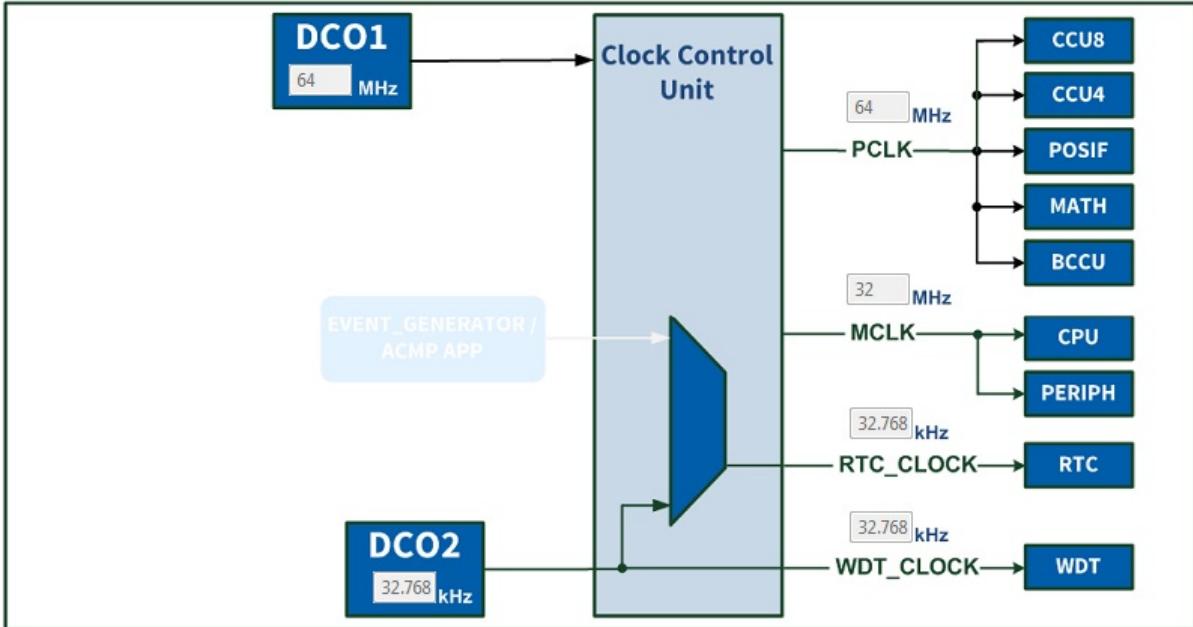
Home

## APP Configuration Parameters

### App Configuration Parameters

### XMC11/XMC12/XMC13 Device

Clock Control Settings | General Settings | Event Settings



### XMC14 Device

Clock Control Settings | External Clock Settings | General Settings | Event Settings

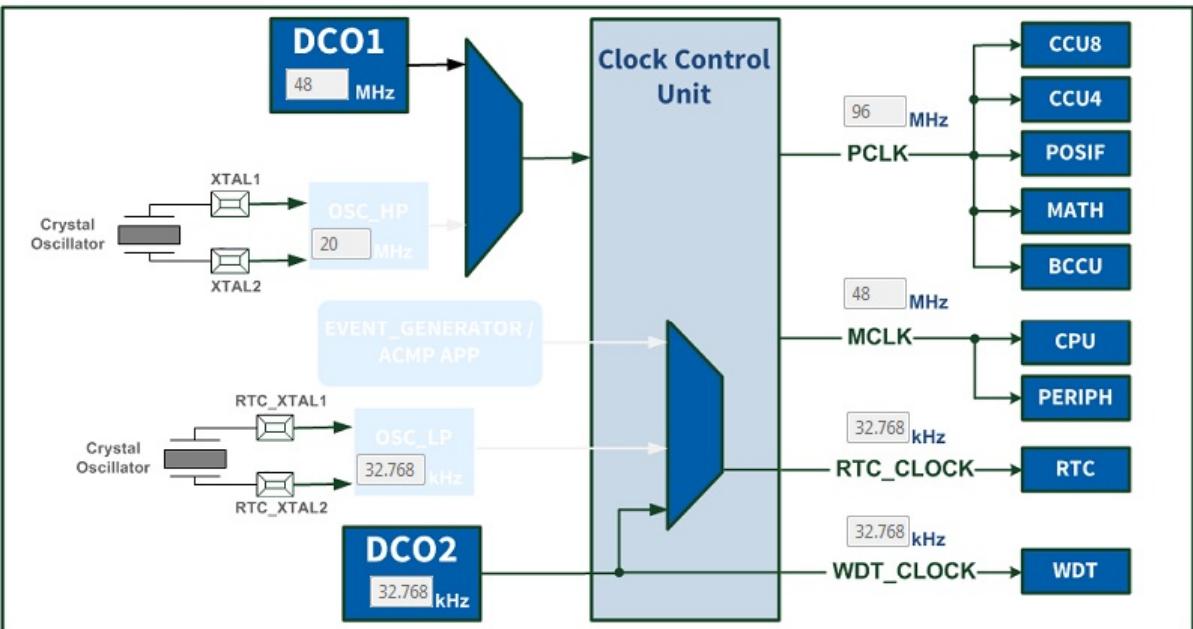


Figure 1: Clock Control Settings



**XMC14 Device**

---

Clock Control Settings External Clock Settings General Settings Event Settings

Enable High Precision Oscillator (OSC\_HP)

OSC\_HP clock source: External Crystal ▾

OSC\_HP clock frequency [MHz]: 20

Enable Low Precision Oscillator (OSC\_LP)

---

**Figure 2: External Clock Settings**



**XMC11/XMC12/XMC13 Device**

General Settings

Clock control unit source:	DCO1	Actual setting [MHz]:	32
Main clock (MCLK) [MHz]:	32.0	Actual setting [MHz]:	32
Fast peripheral clock (PCLK) [MHz]:	2 x MCLK	Actual setting [MHz]:	64
RTC clock source:	DCO2		

**XMC14 Device**

General Settings

Clock control unit source:	DCO1	OSC_HP clock frequency [MHz]	20
Main clock (MCLK) [MHz]:	15.0	Actual setting [MHz]:	15.004
Fast peripheral clock (PCLK) [MHz]:	2 x MCLK	Actual setting [MHz]:	30.007
RTC clock source:	DCO2		
<input type="checkbox"/> Enable DCO1 calibration			
DCO1 calibration source	External Crystal Oscillator (OSC_HP)		

**Figure 3: General Settings**

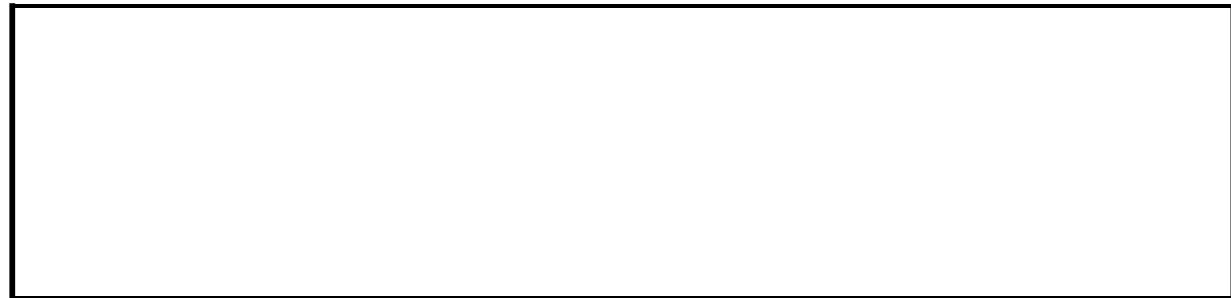
**XMC11/XMC12/XMC13 Device**

Clock Control Settings	General Settings	Event Settings
<input type="checkbox"/> Enable loss of DCO1 clock event	Event handler: UserCallback1	
<input type="checkbox"/> Enable standby clock failure event	Event handler: UserCallback2	

**XMC14 Device**

Clock Control Settings	External Clock Settings	General Settings	Event Settings
<input type="checkbox"/> Enable loss of DCO1 clock event	Event handler: UserCallback1		
<input type="checkbox"/> Enable standby clock failure event	Event handler: UserCallback2		
<input type="checkbox"/> Enable loss of external crystal oscillator clock event	Event handler: UserCallback3		
<input type="checkbox"/> Enable DCO1 out of synchronization event	Event handler: UserCallback4		

**Figure 4: Event Settings**



# CLOCK\_XMC1

Home

## Enumerations

```
CLOCK_XMC1_STATUS {  
enum CLOCK_XMC1_STATUS_SUCCESS = 0U,  
CLOCK_XMC1_STATUS_FAILURE = 1U }
```

## Enumeration Type Documentation

### enum **CLOCK\_XMC1\_STATUS**

#### Enumerator:

*CLOCK\_XMC1\_STATUS\_SUCCESS* APP initialization is success

*CLOCK\_XMC1\_STATUS\_FAILURE* APP initialization is failure

Definition at line [102](#) of file **CLOCK\_XMC1.h**.

---

---

# CLOCK\_XMC1

Home

Data Structures

## Data structures

## Data Structures

---

struct **CLOCK\_XMC1**

Configuration structure for  
**CLOCK\_XMC1** APP. More...

typedef struct **CLOCK\_XMC1** **CLOCK\_XMC1\_t**

Configuration structure for  
**CLOCK\_XMC1** APP.

---



# CLOCK\_XMC1

Home

## Methods

DAVE\_APP\_VERSION\_t **CLOCK\_XMC1\_GetAppVersion** (void)  
Get **CLOCK\_XMC1** APP version.

CLOCK\_XMC1\_STATUS\_t **CLOCK\_XMC1\_Init** (**CLOCK\_XMC1\_t** \*ha)  
Initializes a **CLOCK\_XMC1** APP instance.

void **CLOCK\_XMC1\_SetMCLKFrequency** (uint  
freq\_khz)  
API for ramping up/down the system clock  
frequency.

uint32\_t **OSCHP\_GetFrequency** (void)  
This is a non-weak function, which retrieves  
precision external oscillator frequency.  
**Note:** This function is used by xmc1\_scu  
for internal operations. Therefore the user  
not required to call this API explicitly.

bool **CLOCK\_XMC1\_IsDCO1ExtRefCalibration**  
(void)  
API to check whether DCO1 is synchronize  
the XTAL frequency.

## Methods

## Function Documentation

**DAVE\_APP\_VERSION\_t CLOCK\_XMC1\_GetAppVersion ( void )**

Get **CLOCK\_XMC1** APP version.

**Returns:**

DAVE\_APP\_VERSION\_t APP version information (major, minor and patch number)

**Description:**

The function can be used to check application software compatibility with a specific version of the APP.

Example Usage:

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    DAVE_APP_VERSION_t version;

    // Initialize CLOCK_XMC1 APP:
    // SystemCoreClockSetup() is called from SystemInit().
    init_status = DAVE_Init();
    if(DAVE_STATUS_SUCCESS == init_status)
    {
        version = CLOCK_XMC1_GetAppVersion();
        if (version.major != 4U) {
            // Probably, not the right version.
        }

        // More code here
        while(1) {
```

```
        }
    }
    return (1);
}
```

Definition at line 75 of file [CLOCK\\_XMC1.c](#).

## **CLOCK\_XMC1\_STATUS\_t CLOCK\_XMC1\_Init (CLOCK\_XMC1\_t \* h**

Initializes a [CLOCK\\_XMC1](#) APP instance.

**Parameters:**

**handle** address of [CLOCK\\_XMC1](#) APP handler

**Returns:**

[CLOCK\\_XMC1\\_STATUS\\_SUCCESS](#) : if initialization is successful

[CLOCK\\_XMC1\\_STATUS\\_FAILURE](#) : if initialization is failed

**Description:**

[CLOCK\\_XMC1\\_Init](#) API is called during initialization of DAVE APPS. This API Initializes GLOBAL\_SCU\_XMC1 APP for setting the interrupts and user callback registration.

**Example Usage:**

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t status;

    status = DAVE_Init(); // CLOCK_XMC1_Init API
    // is called during initialization of DAVE APPS
```

```
if(DAVE_STATUS_SUCCESS == status)
{
    // user code

    while(1)
    {

    }
}

return (1);
}
```

Definition at line **88** of file [CLOCK\\_XMC1.c](#).

References [CLOCK\\_XMC1\\_STATUS\\_SUCCESS](#), and  
[CLOCK\\_XMC1::init\\_status](#).

## bool [CLOCK\\_XMC1\\_IsDCO1ExtRefCalibrationReady \( void \)](#)

API to check whether DCO1 is synchronized to the XTAL frequency.

**Parameters:**

**none**

**Returns:**

bool

true : if DCO1 is synchronized to the XTAL frequency

false : if DCO1 is not synchronized to the XTAL frequency

**Description:**

The function can be used to check whether DCO1 is synchronized to the XTAL frequency.

Example Usage:

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    #if(CLOCK_XMC1_DC01_CALIBRATION_ENABLED)
    bool is_synchronized = false;
    #endif
    // Initialize CLOCK_XMC1 APP:
    // SystemCoreClockSetup() is called from SystemInit().
    init_status = DAVE_Init();
    if(DAVE_STATUS_SUCCESS == init_status)
    {
        // User code here
        #if(CLOCK_XMC1_DC01_CALIBRATION_ENABLED)
            is_synchronized = CLOCK_XMC1_IsDC01ExtRefCalibrationReady(); // check whether DC01 is synchronized

                // to the XTAL frequency or not
            if(is_synchronized == true)
            {
                // User code here
                // Do baud rate configuration related to communication protocol
                // start PWM in compare mode
                // start RTC in the RTC domain

            }
        #endif
        // More code here
        while(1) {

        }
    }
    return (1);
}
```

```
}
```

Definition at line **164** of file **CLOCK\_XMC1.c**.

**void CLOCK\_XMC1\_SetMCLKFrequency ( uint32\_t freq\_khz )**

API for ramping up/down the system clock frequency.

**Parameters:**

**target\_freq** required frequency in Hz.

**Returns:**

none

**Description:**

The function can be used for ramping up/down the system clock frequency.

Example Usage:

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    uint32_t freq_khz = 1000U; // 1MHz is the target frequency
    // Initialize CLOCK_XMC1 APP:
    // SystemCoreClockSetup() is called from SystemInit().
    init_status = DAVE_Init();
    if(DAVE_STATUS_SUCCESS == init_status)
    {
        CLOCK_XMC1_SetMCLKFrequency(freq_khz); // sy
```

```
stem clock frequency is ramping down to 1 MHz
// More code here
while(1) {

}

return (1);
}
```

Definition at line [149](#) of file **CLOCK\_XMC1.c**.

### **uint32\_t OSCHP\_GetFrequency ( void )**

This is a non-weak function, which retrieves high precision external oscillator frequency.

**Note: This function is used by xmc1\_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.**

**Returns:**

uint32\_t Range: 4 to 20 in External Crystal Mode / External Direct Input Mode.

**Description:**

This function to retrieves the external high precision oscillator frequency value, derived from either "External Crystal Mode" or "External Direct Input Mode"

Definition at line [156](#) of file **CLOCK\_XMC1.c**.

# CLOCK\_XMC1

Home

## Usage

### Usage

**CLOCK\_XMC1** is a *leaf level APP*, and consumed by all use-case APPs such as : *UART, PWM, SPI, CAN (top level) APPs.*

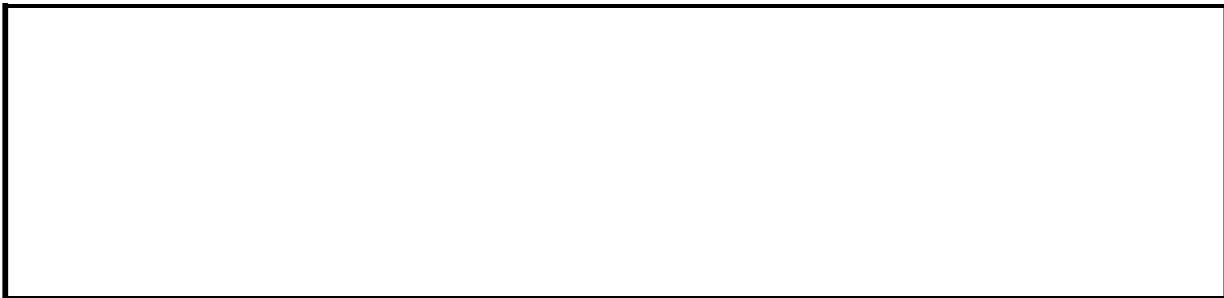


# CLOCK\_XMC1

Home

## Release History

### Release History



# CLOCK\_XMC1

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

## Data Structures

Here are the data structures with brief descriptions:

**CLOCK\_XMC1** Configuration structure for **CLOCK\_XMC1 APP**



# CLOCK\_XMC1

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

## CLOCK\_XMC1 Struct Reference

[Data structures](#)

---

## Detailed Description

Configuration structure for **CLOCK\_XMC1** APP.

Definition at line **123** of file **CLOCK\_XMC1.h**.

```
#include <CLOCK_XMC1.h>
```

## Data Fields

bool **init\_status**

---

## Field Documentation

**bool [CLOCK\\_XMC1::init\\_status](#)**

APP is initialized or not.

Definition at line [146](#) of file [CLOCK\\_XMC1.h](#).

Referenced by [CLOCK\\_XMC1\\_Init\(\)](#).

---

The documentation for this struct was generated from the following file:

- [CLOCK\\_XMC1.h](#)
- 



# CLOCK\_XMC1

Home

Data Structures

Data Structure Index

Data Fields

## Data Structure Index

C

C

[CLOCK\\_XMC1](#)

C



# CLOCK\_XMC1

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- init\_status : [CLOCK\\_XMC1](#)
-

# CLOCK\_XMC1

Home		
Data Structures	Data Structure Index	Data Fields
All	Variables	

- init\_status : CLOCK\_XMC1
-

# CLOCK\_XMC1

Home

File List

Globals

## File List

Here is a list of all documented files with brief descriptions:

[CLOCK\\_XMC1.c \[code\]](#)

[CLOCK\\_XMC1.h \[code\]](#)

# CLOCK\_XMC1

[Home](#)

[File List](#)

[Globals](#)

[Functions](#)

## CLOCK\_XMC1.c File Reference

## Detailed Description

**Date:**

2015-05-04

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file [\*\*CLOCK\\_XMC1.c\*\*](#).

```
#include "clock_xmc1.h"
```

## Functions

DAVE_APP_VERSION_t	<b>CLOCK_XMC1_GetAppVersion</b> (void) Get <b>CLOCK_XMC1</b> APP version.
CLOCK_XMC1_STATUS_t	<b>CLOCK_XMC1_Init</b> ( <b>CLOCK_XMC1_t</b> *ha) Initializes a <b>CLOCK_XMC1</b> APP instance.
void	<b>CLOCK_XMC1_SetMCLKFrequency</b> (uint freq_khz) API for ramping up/down the system clock frequency.
uint32_t	<b>OSCHP_GetFrequency</b> (void) This is a non-weak function, which retrieves precision external oscillator frequency. <b>Note:</b> This function is used by xmc1_scu for internal operations. Therefore the user not required to call this API explicitly.
bool	<b>CLOCK_XMC1_IsDCO1ExtRefCalibration</b> (void) API to check whether DCO1 is synchronize the XTAL frequency.

## Function Documentation

**CLOCK\_XMC1\_STATUS\_t CLOCK\_XMC1\_Init( CLOCK\_XMC1\_t \* h**

Initializes a **CLOCK\_XMC1** APP instance.

**Parameters:**

**handle** address of **CLOCK\_XMC1** APP handler

**Returns:**

**CLOCK\_XMC1\_STATUS\_SUCCESS** : if initialization is successful

**CLOCK\_XMC1\_STATUS\_FAILURE** : if initialization is failed

**Description:**

**CLOCK\_XMC1\_Init** API is called during initialization of DAVE APPS. This API Initializes GLOBAL\_SCU\_XMC1 APP for setting the interrupts and user callback registration.

**Example Usage:**

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t status;

    status = DAVE_Init(); // CLOCK_XMC1_Init API
    is called during initialization of DAVE APPS
    if(DAVE_STATUS_SUCCESS == status)
    {
        // user code

        while(1)
        {
```

```
    }
}
return (1);
}
```

Definition at line **88** of file [CLOCK\\_XMC1.c](#).

References [CLOCK\\_XMC1\\_STATUS\\_SUCCESS](#), and  
[CLOCK\\_XMC1::init\\_status](#).

## **bool CLOCK\_XMC1\_IsDCO1ExtRefCalibrationReady ( void )**

API to check whether DCO1 is synchronized to the XTAL frequency.

### **Parameters:**

**none**

### **Returns:**

**bool**

true : if DCO1 is synchronized to the XTAL frequency

false : if DCO1 is not synchronized to the XTAL frequency

### **Description:**

The function can be used to check whether DCO1 is synchronized to the XTAL frequency.

### **Example Usage:**

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    #if(CLOCK_XMC1_DCO1_CALIBRATION_ENABLED)
    bool is_synchronized = false;
```

```
#endif
// Initialize CLOCK_XMC1 APP:
// SystemCoreClockSetup() is called from SystemInit().
init_status = DAVE_Init();
if(DAVE_STATUS_SUCCESS == init_status)
{
    // User code here
#if(CLOCK_XMC1_DC01_CALIBRATION_ENABLED)
    is_synchronized = CLOCK_XMC1_IsDC01ExtRefCalibrationReady(); // check whether DC01 is synchronized

        // to the XTAL frequency or not
    if(is_synchronized == true)
    {
        // User code here
        // Do baud rate configuration related to communication protocol
        // start PWM in compare mode
        // start RTC in the RTC domain

    }
#endif
    // More code here
    while(1) {

    }
}
return (1);
}
```

Definition at line [164](#) of file **CLOCK\_XMC1.c**.

## **void CLOCK\_XMC1\_SetMCLKFrequency ( uint32\_t freq\_khz )**

API for ramping up/down the system clock frequency.

**Parameters:**

**target\_freq** required frequency in Hz.

**Returns:**

none

**Description:**

The function can be used for ramping up/down the system clock frequency.

Example Usage:

```
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    uint32_t freq_khz = 1000U; // 1MHz is the target frequency
    // Initialize CLOCK_XMC1 APP:
    // SystemCoreClockSetup() is called from SystemInit().
    init_status = DAVE_Init();
    if(DAVE_STATUS_SUCCESS == init_status)
    {
        CLOCK_XMC1_SetMCLKFrequency(freq_khz); // system clock frequency is ramping down to 1 MHz
        // More code here
        while(1) {

        }
    }
    return (1);
}
```

```
}
```

Definition at line [149](#) of file **CLOCK\_XMC1.c**.

### **uint32\_t OSCHP\_GetFrequency ( void )**

This is a non-weak function, which retrieves high precision external oscillator frequency.

**Note: This function is used by xmc1\_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.**

**Returns:**

uint32\_t Range: 4 to 20 in External Crystal Mode / External Direct Input Mode.

**Description:**

This function to retrieves the external high precision oscillator frequency value, derived from either "External Crystal Mode" or "External Direct Input Mode"

Definition at line [156](#) of file **CLOCK\_XMC1.c**.

Go to the source code of this file.

# CLOCK\_XMC1

[Home](#)

[File List](#)

[Globals](#)

[Data Structures](#)

## CLOCK\_XMC1.h File Reference

## Detailed Description

**Date:**

2015-06-20

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file [\*\*CLOCK\\_XMC1.h\*\*](#).

```
#include <xmc_scu.h> #include <DAVE_common.h>
#include "clock_xmc1_conf.h"
#include "clock_xmc1_extern.h"
```

## Data Structures

---

struct **CLOCK\_XMC1**

Configuration structure for **CLOCK\_XMC1** APP. More...

## TypeDefs

```
typedef struct CLOCK_XMC1 CLOCK_XMC1_t
{
    /* Configuration structure for
       CLOCK_XMC1 APP.
}
```

## Functions

DAVE_APP_VERSION_t	<b>CLOCK_XMC1_GetAppVersion</b> (void) Get <b>CLOCK_XMC1</b> APP version.
CLOCK_XMC1_STATUS_t	<b>CLOCK_XMC1_Init</b> ( <b>CLOCK_XMC1_t</b> *ha) Initializes a <b>CLOCK_XMC1</b> APP instance.
void	<b>CLOCK_XMC1_SetMCLKFrequency</b> (uint freq_khz) API for ramping up/down the system clock frequency.
uint32_t	<b>OSCHP_GetFrequency</b> (void) This is a non-weak function, which retrieves precision external oscillator frequency. <b>Note:</b> This function is used by xmc1_scu for internal operations. Therefore the user not required to call this API explicitly.
bool	<b>CLOCK_XMC1_IsDCO1ExtRefCalibration</b> (void) API to check whether DCO1 is synchronize the XTAL frequency.
enum	<b>CLOCK_XMC1_STATUS</b> { <b>CLOCK_XMC1_STATUS_SUCCESS</b> = 0U <b>CLOCK_XMC1_STATUS_FAILURE</b> = 1U }

Go to the source code of this file.

# CLOCK\_XMC1

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- CLOCK\_XMC1\_GetAppVersion() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)
- CLOCK\_XMC1\_Init() : [CLOCK\\_XMC1.h](#) , [CLOCK\\_XMC1.c](#)
- CLOCK\_XMC1\_IsDCO1ExtRefCalibrationReady() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)
- CLOCK\_XMC1\_SetMCLKFrequency() : [CLOCK\\_XMC1.h](#) , [CLOCK\\_XMC1.c](#)
- CLOCK\_XMC1\_STATUS : [CLOCK\\_XMC1.h](#)
- CLOCK\_XMC1\_STATUS\_FAILURE : [CLOCK\\_XMC1.h](#)
- CLOCK\_XMC1\_STATUS\_SUCCESS : [CLOCK\\_XMC1.h](#)
- CLOCK\_XMC1\_t : [CLOCK\\_XMC1.h](#)
- OSCHP\_GetFrequency() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)

# CLOCK\_XMC1

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- CLOCK\_XMC1\_GetAppVersion() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)
  - CLOCK\_XMC1\_Init() : [CLOCK\\_XMC1.h](#) , [CLOCK\\_XMC1.c](#)
  - CLOCK\_XMC1\_IsDCO1ExtRefCalibrationReady() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)
  - CLOCK\_XMC1\_SetMCLKFrequency() : [CLOCK\\_XMC1.h](#) , [CLOCK\\_XMC1.c](#)
  - OSCHP\_GetFrequency() : [CLOCK\\_XMC1.c](#) , [CLOCK\\_XMC1.h](#)
-

# CLOCK\_XMC1

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- CLOCK\_XMC1\_t : [CLOCK\\_XMC1.h](#)
- 



# CLOCK\_XMC1

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- CLOCK\_XMC1\_STATUS : [CLOCK\\_XMC1.h](#)
-

# CLOCK\_XMC1

Home					
File List	Globals				
All	Functions	Typedefs	Enumerations	Enumerator	

- CLOCK\_XMC1\_STATUS\_FAILURE : [CLOCK\\_XMC1.h](#)
  - CLOCK\_XMC1\_STATUS\_SUCCESS : [CLOCK\\_XMC1.h](#)
-

# CLOCK\_XMC1

Home		
File List	Globals	

# CLOCK\_XMC1.h

[Go to the documentation of this file.](#)

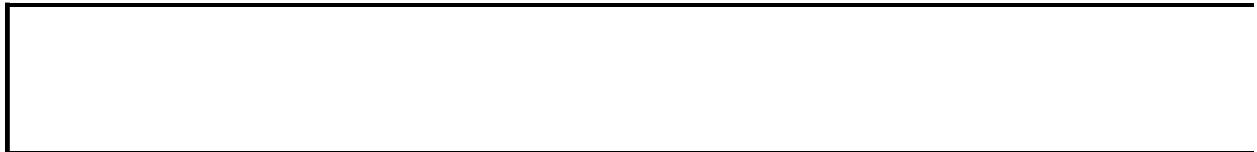
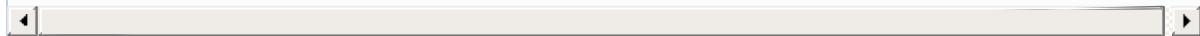
```
00001
00059 #ifndef CLOCK_XMC1_H
00060 #define CLOCK_XMC1_H
00061
00062 /*****
00063 **** * HEADER FILES
00064 **** */
00065 #include <xmc_scu.h>
00066 #include <DAVE_common.h>
00067 #include "clock_xmc1_conf.h"
00068 #ifdef CLOCK_XMC1_INTERRUPT_ENABLED
00069 #include "../GLOBAL_SCU_XMC1/global_scu_xmc1
00070 .h"
00071 #endif
00072 /*****
00073 **** * MACROS
00074 **** */
00075 /*****
```

```
00074 #if (((XMC_LIB_MAJOR_VERSION == 2U) && \
00075             (XMC_LIB_MINOR_VERSION >= 0U) && \
00076             (XMC_LIB_PATCH_VERSION >= 0U)))
00077 #error "CLOCK_XMC1 requires XMC Peripheral L
ibrary v2.0.0 or higher"
00078 #endif
00079
00092 /*****
***** * ENUMS
00094 *****
***** */
00099 /*
00100 * @brief enumeration for CLOCK_XMC1 APP
00101 */
00102 typedef enum CLOCK_XMC1_STATUS
00103 {
00104     CLOCK_XMC1_STATUS_SUCCESS = 0U,
00105     CLOCK_XMC1_STATUS_FAILURE = 1U
00106 } CLOCK_XMC1_STATUS_t;
00107
00112 /*****
***** * DATA STRUCTURES
00114 *****
***** */
00113 /*
00114 * @brief structure for CLOCK_XMC1
00115 */
00116
00117 #ifdef CLOCK_XMC1_INTERRUPT_ENABLED
00118
00119     GLOBAL_SCU_XMC1_t *const global_scu_handle
00120     ptr;
00121
00122 #ifdef CLOCK_XMC1_LOCI_EVENT_ENABLED
```

```
00129     void (*callback_function_loci)(void);
00130 #endif
00131 #ifdef CLOCK_XMC1_STDBYCLKFAIL_EVENT_ENABLED
00132     void (*callback_function_stdbyclkfail)(voi
d);
00133 #endif
00134
00135 #if (UC_SERIES == XMC14)
00136
00137 #ifdef CLOCK_XMC1_LOSS_EXT_CLOCK_EVENT_ENABL
ED
00138     void (*callback_function_loss_ext_clock)(v
oid);
00139 #endif
00140 #ifdef CLOCK_XMC1_DC01_OUT_SYNC_EVENT_ENABLED

00141     void (*callback_function_dc01_out_sync)(vo
id);
00142 #endif
00143 #endif
00144
00145 #endif
00146     bool init_status;
00147 } CLOCK_XMC1_t;
00148
00153 #ifdef __cplusplus
00154 extern "C" {
00155 #endif
00156 /*****
00157 ****
00158 * API Prototypes
00159 ****
00160 ****
00161 ****
00162 ****
00163 ****
00164 ****
00165 ****
00166 ****
00167 ****
00168 ****
00169 ****
00170 ****
00171 ****
00172 ****
00173 ****
00174 ****
00175 ****
00176 ****
00177 ****
00178 ****
00179 ****
00180 ****
00181 ****
00182 ****
00183 ****
00184 ****
00185 ****
00186 ****
00187 ****
00188 ****
00189 ****
00190 ****
00191 ****
00192 ****
00193 ****
00194 ****
00195 ****
00196 ****
00197 ****
00198 ****
00199 DAVE_APP_VERSION_t CLOCK_XMC1_GetAppVersion(
void);
```

```
00200
00236 CLOCK_XMC1_STATUS_t CLOCK_XMC1_Init(CLOCK_XM
C1_t *handle);
00237
00270 void CLOCK_XMC1_SetMCLKFrequency(uint32_t fr
eq_khz);
00271
00272 #if (CLOCK_XMC1_OSCHP_ENABLED)
00273
00285 uint32_t OSCHP_GetFrequency(void);
00286 #endif
00287
00288 #if (CLOCK_XMC1_DC01_CALIBRATION_ENABLED)
00289
00337 bool CLOCK_XMC1_IsDC01ExtRefCalibrationReady(
void);
00338 #endif
00339
00343 #ifdef __cplusplus
00344 }
00345 #endif
00346
00347 #include"clock_xmc1_extern.h"
00348
00349 #endif /* End of _CLOCK_XMC1_H_ */
```



# CLOCK\_XMC1

Home

File List

Globals

## CLOCK\_XMC1.c

Go to the documentation of this file.

```
00001
00054 /*****
00055 ****
00055 ****
00055 * HEADER FILES
00056 ****
00056 ****
00056 ****
00057 #include "clock_xmc1.h"
00058
00059 /*****
00059 ****
00059 ****
00059 ****
00060 * MACROS
00061 ****
00061 ****
00061 ****
00062
00063 /*****
00063 ****
00063 ****
00064 * LOCAL DATA
00065 ****
00065 ****
00065 ****
```

```
00066
00067 /*****
00068 ****
00069 * LOCAL ROUTINES
00070 ****
00071 ****
00072 * API IMPLEMENTATION
00073 ****
00074 /* API to retrieve version of the APP */
00075 DAVE_APP_VERSION_t CLOCK_XMC1_GetAppVersion(
void)
00076 {
00077     DAVE_APP_VERSION_t version;
00078     version.major = (uint8_t)CLOCK_XMC1_MAJOR_
VERSION;
00079     version.minor = (uint8_t)CLOCK_XMC1_MINOR_
VERSION;
00080     version.patch = (uint8_t)CLOCK_XMC1_PATCH_
VERSION;
00081
00082     return (version);
00083 }
00084
00085 /*
00086 * API to initialize the CLOCK_XMC1 APP Inte
rrupts
00087 */
00088 CLOCK_XMC1_STATUS_t CLOCK_XMC1_Init(CLOCK_XM
C1_t *handle)
```

```
00089 {
00090     CLOCK_XMC1_STATUS_t status = CLOCK_XMC1_ST
ATUS_SUCCESS;
00091     CLOCK_XMC1_STATUS_t loci_event_status = CL
OCK_XMC1_STATUS_SUCCESS;
00092     CLOCK_XMC1_STATUS_t stdbyclkfail_status =
CLOCK_XMC1_STATUS_SUCCESS;
00093     CLOCK_XMC1_STATUS_t loss_ext_clock_event_s
tatus = CLOCK_XMC1_STATUS_SUCCESS;
00094     CLOCK_XMC1_STATUS_t dco1_out_sync_status =
CLOCK_XMC1_STATUS_SUCCESS;
00095     if (handle->init_status == false)
00096     {
00097 #ifdef CLOCK_XMC1_INTERRUPT_ENABLED
00098
00099     status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU
_XMC1_Init(handle->global_scu_handleptr);
00100     if (CLOCK_XMC1_STATUS_SUCCESS == status)
00101     {
00102 #ifdef CLOCK_XMC1_LOCI_EVENT_ENABLED
00103         /* Initialization of CPU_CTRL_XMC1 APP
 */
00104         loci_event_status = (CLOCK_XMC1_STATUS
_t)GLOBAL_SCU_XMC1_RegisterCallback(
00105                         GLOBAL_SCU_XMC1_E
VENT_LOCI, handle->callback_function_loci);
00106         /* Enable Loss of DC01 Clock Event */
00107         XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_S
CU_XMC1_EVENT_LOCI);
00108 #endif
00109 #ifdef CLOCK_XMC1_STDBYCLKFAIL_EVENT_ENABLED
00110         /* Initialization of CPU_CTRL_XMC1 APP
 */
00111         stdbyclkfail_status = (CLOCK_XMC1_STAT
US_t)GLOBAL_SCU_XMC1_RegisterCallback(
00112                         GLOBAL_SCU_XMC1
_EVENT_STDBYCLKFAIL, handle->callback_function_std
```

```

byclkfail);
0013      /* Enable Standby Clock Failure Event
*/
0014      XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_STDBYCLKFAIL);
0015 #endif
0016
0017 #if (UC_SERIES == XMC14)
0018
0019 #ifdef CLOCK_XMC1_LOSS_EXT_CLOCK_EVENT_ENABLED
0020      /* Initialization of CPU_CTRL_XMC1 APP
*/
0021      loss_ext_clock_event_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(
0022                                              GLOBAL_SCU_XMC1_EVENT_LOSS_EXT_CLOCK, handle->callback_function_loss_ext_clock);
0023      /* Enable Loss of external OSC_HP clock Event */
0024      XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_LOSS_EXT_CLOCK);
0025 #endif
0026 #ifdef CLOCK_XMC1_DC01_OUT_SYNC_EVENT_ENABLED

0027      /* Initialization of CPU_CTRL_XMC1 APP
*/
0028      dc01_out_sync_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(
0029                                              GLOBAL_SCU_XMC1_EVENT_DC01_OUT_SYNC, handle->callback_function_dc01_out_sync);
0030      /* Enable DC01 Out of SYNC Event */
0031      XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_DC01_OUT_SYNC);
0032 #endif
0033

```

```
00134 #endif
00135     }
00136
00137 #endif
00138     status = (CLOCK_XMC1_STATUS_t) (((uint32_
t)loci_event_status) | ((uint32_t)stdbyclkfail_sta
tus) |
00139                                     ((uin
t32_t)loss_ext_clock_event_status) | ((uint32_t)dc
o1_out_sync_status));
00140     if (CLOCK_XMC1_STATUS_SUCCESS == status)
00141     {
00142         handle->init_status = true;
00143     }
00144 }
00145 return (status);
00146 }
00147
00148 /* API for ramping up/down the system clock
frequency */
00149 void CLOCK_XMC1_SetMCLKFrequency(uint32_t fr
eq_khz)
00150 {
00151     XMC_SCU_CLOCK_SetMCLKFrequency(freq_khz);
00152 }
00153
00154 #if (CLOCK_XMC1_OSCHP_ENABLED)
00155 /* API to retrieve high precision external
oscillator frequency */
00156 uint32_t OSCHP_GetFrequency(void)
00157 {
00158     return (CLOCK_XMC1_OSCHP_FREQUENCY);
00159 }
00160#endif
00161
00162 #if (CLOCK_XMC1_DC01_CALIBRATION_ENABLED)
00163 /* API to check whether DC01 is synchronize
```

```
d to the XTAL frequency */
00164 bool CLOCK_XMC1_IsDC01ExtRefCalibrationReady(
void)
00165 {
00166     return (XMC_SCU_CLOCK_IsDC01ExtRefCalibrat
ionReady());
00167 }
00168 #endif
```

