# CLOCK_XMC1

<table>
<thead>
<tr>
<th>Home</th>
<th>Apps</th>
</tr>
</thead>
</table>

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
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).
## Abbreviations and Definitions

### Abbreviations:

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

### Definitions:
<table>
<thead>
<tr>
<th>Singleton</th>
<th>Only single instance of the APP is permitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharable</td>
<td>Resource sharing with other APPs is permitted</td>
</tr>
<tr>
<td>initProvider</td>
<td>Provides the initialization routine</td>
</tr>
<tr>
<td>Physical connectivity</td>
<td>Hardware inter/intra peripheral (constant) signal connection</td>
</tr>
<tr>
<td>Conditional connectivity</td>
<td>Constrained hardware inter/intra peripheral signal connection</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Indicates consumption of low level (dependent) DAVE APPs</td>
</tr>
</tbody>
</table>
CLOCK_XMC1

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:

\[
MCLK = \frac{dco\_clk}{2\times(IDIV + FDIV/256)} \text{ for } IDIV>0
\]

Here dco_dclk 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:

\[
MCLK = \frac{DCLK}{2\times(IDIV + FDIV/1024)} \text{ for } IDIV>0
\]

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.
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

Limitations

None
CLOCK_XMC1

Architecture Description

Figure 1 explains the architecture of the 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 I0 signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Input/Output</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk_rtc</td>
<td>Input</td>
<td>Conditionally</td>
<td>RTC clock input selection signal. Can be connected with ERU or ANCMP APPs</td>
</tr>
</tbody>
</table>
CLOCK_XMC1

<table>
<thead>
<tr>
<th>Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP Configuration Parameters</td>
</tr>
</tbody>
</table>

App Configuration Parameters
Figure 1: Clock Control Settings
Figure 2: External Clock Settings
Figure 3: General Settings
Figure 4: Event Settings
CLOCK_XMC1

Enumerations

<table>
<thead>
<tr>
<th>enum</th>
<th>CLOCK_XMC1_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLOCK_XMC1_STATUS_SUCCESS = 0U,</td>
</tr>
<tr>
<td></td>
<td>CLOCK_XMC1_STATUS_FAILURE = 1U }</td>
</tr>
</tbody>
</table>
**Enumeration Type Documentation**

**enum CLOCK_XMC1_STATUS**

**Enumerator:**

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOCK_XMC1_STATUS_SUCCESS</td>
<td>APP initialization is success</td>
</tr>
<tr>
<td>CLOCK_XMC1_STATUS_FAILURE</td>
<td>APP initialization is failure</td>
</tr>
</tbody>
</table>

Definition at line 102 of file CLOCK_XMC1.h.
<table>
<thead>
<tr>
<th>Home</th>
<th>Data Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data structures</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Data Structures

```plaintext
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

## Methods

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC1_GetAppVersion (void)</td>
<td>Get CLOCK_XMC1 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC1_STATUS_t</td>
<td>CLOCK_XMC1_Init (CLOCK_XMC1_t *handle)</td>
<td>Initializes a CLOCK_XMC1 APP instance.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC1_SetMCLKFrequency (uint32_t freq_khz)</td>
<td>API for ramping up/down the system clock frequency.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>OSCHP_GetFrequency (void)</td>
<td>This is a non-weak function, which retrieves precision external oscillator frequency.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: This function is used by xmc1_scu for internal operations. Therefore the user is not required to call this API explicitly.</td>
</tr>
<tr>
<td>bool</td>
<td>CLOCK_XMC1_IsDCO1ExtRefCalibrationReady (void)</td>
<td>API to check whether DCO1 is synchronized with the XTAL frequency.</td>
</tr>
</tbody>
</table>
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) {
```
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:**

```c
#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_DCO1_CALIBRATION_ENABLED)
        is_synchronized = CLOCK_XMC1_IsDCO1ExtRefCalibrationReady(); // check whether DCO1 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);
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:**

```c
#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);
}

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 149 of file CLOCK_XMC1.c.
CLOCK_XMC1

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

#### Release History

<table>
<thead>
<tr>
<th>Release History</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

---
## CLOCK_XMC1

### Data Structures

Here are the data structures with brief descriptions:

<table>
<thead>
<tr>
<th>CLOCK_XMC1</th>
<th>Configuration structure for CLOCK_XMC1 APP</th>
</tr>
</thead>
</table>

Detailed Description

Configuration structure for `CLOCK_XMC1` APP.

Definition at line 123 of file `CLOCK_XMC1.h`.

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

<table>
<thead>
<tr>
<th>bool</th>
<th>init_status</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Data Structure Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

CLOCK_XMC1

C
CLOCK_XMC1

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

- **init_status**: `CLOCK_XMC1`
CLOCK_XMC1

**File List**

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

- **CLOCK_XMC1.c**
- **CLOCK_XMC1.h**
# CLOCK_XMC1

## CLOCK_XMC1.c File Reference

<table>
<thead>
<tr>
<th>Home</th>
<th>File List</th>
<th>Globals</th>
<th>Functions</th>
</tr>
</thead>
</table>

---
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC1_GetAppVersion (void) Get CLOCK_XMC1 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC1_STATUS_t</td>
<td>CLOCK_XMC1_Init (CLOCK_XMC1_t *handle) Initializes a CLOCK_XMC1 APP instance.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC1_SetMCLKFrequency (uint freq_khz) API for ramping up/down the system clock frequency.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>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 is not required to call this API explicitly.</td>
</tr>
<tr>
<td>bool</td>
<td>CLOCK_XMC1_IsDCO1ExtRefCalibrationReady (void) API to check whether DCO1 is synchronize the XTAL frequency.</td>
</tr>
</tbody>
</table>
Function Documentation

CLOCK_XMC1_STATUS_t CLOCK_XMC1_Init ( CLOCK_XMC1_t * handle)

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:

```c
#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)
        {
```
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:

```c
#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_DCO1_CALIBRATION_ENABLED)
            is_synchronized = CLOCK_XMC1_IsDCO1ExtRefCalibrationReady(); // check whether DCO1 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);
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 Syste
   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.

```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

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_xmc1Extern.h"
Data Structures

struct CLOCK_XMC1
Configuration structure for CLOCK_XMC1 APP. More...
typedef struct \texttt{CLOCK\_XMC1} \texttt{CLOCK\_XMC1\_t}
Configuration structure for \texttt{CLOCK\_XMC1} APP.
### Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC1_GetAppVersion (void)</td>
<td>Get CLOCK_XMC1 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC1_STATUS_t</td>
<td>CLOCK_XMC1_Init (CLOCK_XMC1_t *handle)</td>
<td>Initializes a CLOCK_XMC1 APP instance.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC1_SetMCLKFrequency (uint32_t freq_khz)</td>
<td>API for ramping up/down the system clock frequency.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>OSCHP_GetFrequency (void)</td>
<td>This is a non-weak function, which retrieves precision external oscillator frequency. <strong>Note:</strong> This function is used by xmc1_scu for internal operations. Therefore the user is not required to call this API explicitly.</td>
</tr>
<tr>
<td>bool</td>
<td>CLOCK_XMC1_IsDCO1ExtRefCalibrationReady (void)</td>
<td>API to check whether DCO1 is synchronized to the XTAL frequency.</td>
</tr>
<tr>
<td>enum</td>
<td>CLOCK_XMC1_STATUS { CLOCK_XMC1_STATUS_SUCCESS = 0U, CLOCK_XMC1_STATUS_FAILURE = 1U }</td>
<td></td>
</tr>
</tbody>
</table>

Go to the source code of this file.
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

- **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

- CLOCK_XMC1_t: CLOCK_XMC1.h
CLOCK_XMC1

- CLOCK_XMC1_STATUS : CLOCK_XMC1.h
CLOCK_XMC1

- CLOCK_XMC1_STATUS_FAILURE: CLOCK_XMC1.h
- CLOCK_XMC1_STATUS_SUCCESS: CLOCK_XMC1.h
CLOCK_XMC1

Go to the documentation of this file.

```
#ifndef CLOCK_XMC1_H
#define CLOCK_XMC1_H

/*******************************************
**************************************************
**************************
*/
#include <xmc_scu.h>
#include <DAVE_common.h>
#include "clock_xmc1_conf.h"
#ifdef CLOCK_XMC1_INTERRUPT_ENABLED
#include "../GLOBAL_SCU_XMC1/global_scu_xmc1.h"
#endif

/******************************************
**************************************************
**************************
*/
#include "../GLOBAL_SCU_XMC1/global_scu_xmc1.h"
#endif

#ifdef CLOCK_XMC1_INTERRUPT_ENABLED
#include "../GLOBAL_SCU_XMC1/global_scu_xmc1.h"
#endif

@
```
```c
#if (!(XMC_LIB_MAJOR_VERSION == 2U) && \n    (XMC_LIB_MINOR_VERSION >= 0U) && \n    (XMC_LIB_PATCH_VERSION >= 0U))
#error "CLOCK_XMC1 requires XMC Peripheral Library v2.0.0 or higher"
#endif

/******************************************
**************************************************
**************************
* ENUMS
**************************************************
*******************************************

/* @brief enumeration for CLOCK_XMC1 APP */
typedef enum CLOCK_XMC1_STATUS {
    CLOCK_XMC1_STATUS_SUCCESS = 0U,
    CLOCK_XMC1_STATUS_FAILURE = 1U
} CLOCK_XMC1_STATUS_t;

/*******************************************
**************************************************
*************************
* DATA STRUCTURES
**************************************************
********************************************

/*****************************/
typedef struct CLOCK_XMC1 {
    #ifdef CLOCK_XMC1_INTERRUPT_ENABLED
        #ifdef CLOCK_XMC1_LOCI_EVENT_ENABLED
GLOBAL_SCU_XMC1_t *const global_scu_handle ptr;
```
void (*callback_function_loci)(void);
#endif

#ifdef CLOCK_XMC1_STDBYCLKFAIL_EVENT_ENABLED
void (*callback_function_stdbyclkfail)(void);
#endif

#if (UC_SERIES == XMC14)
#ifdef CLOCK_XMC1_LOSS_EXT_CLOCK_EVENT_ENABLED
void (*callback_function_loss_ext_clock)(void);
#endif
#ifdef CLOCK_XMC1_DCO1_OUT_SYNC_EVENT_ENABLED
void (*callback_function_dco1_out_sync)(void);
#endif
#endif

bool init_status;
}
CLOCK_XMC1_t;

#ifdef __cplusplus
extern "C" {
#endif

/*******************************************
**************************************************
**************************
* API Prototypes
**************************************************
**************************

DAVE_APP_VERSION_t CLOCK_XMC1_GetAppVersion(void);
CLOCK_XMC1_STATUS_t CLOCK_XMC1_Init(CLOCK_XMC1_t *handle);

void CLOCK_XMC1_SetMCLKFrequency(uint32_t freq_khz);

#if (CLOCK_XMC1_OSCHP_ENABLED)
uint32_t OSCHP_GetFrequency(void);
#endif

#if (CLOCK_XMC1_DCO1_CALIBRATION_ENABLED)
bool CLOCK_XMC1_IsDCO1ExtRefCalibrationReady(void);
#endif

#include "clock_xmc1Extern.h"

/* End of _CLOCK_XMC1_H_ */
CLOCK_XMC1

Go to the documentation of this file.

```
00001
00054  *******************************************
          *******************************************
          *******************************************
00055   * HEADER FILES

00056   *******************************************
          *******************************************
          *******************************************
00057  #include "clock_xmc1.h"
00058
00059  *******************************************
          *******************************************
          *******************************************
00060   * MACROS
00061
00062
00063  *******************************************
          *******************************************
          *******************************************
00064   * LOCAL DATA
00065
00066  *******************************************
```

CLOCK_XMC1.c
/*
 * API to retrieve version of the APP */

DAVE_APP_VERSION_t CLOCK_XMC1_GetAppVersion(void)
{
    DAVE_APP_VERSION_t version;
    version.major = (uint8_t)CLOCK_XMC1_MAJOR_VERSION;
    version.minor = (uint8_t)CLOCK_XMC1_MINOR_VERSION;
    version.patch = (uint8_t)CLOCK_XMC1_PATCH_VERSION;
    return (version);
}

/*
 * API to initialize the CLOCK_XMC1 APP Interrupts */

CLOCK_XMC1_STATUS_t CLOCK_XMC1_Init(CLOCK_XMC1_t *handle)
{  
  CLOCK_XMC1_STATUS_t status = CLOCK_XMC1_STATUS_SUCCESS;
  CLOCK_XMC1_STATUS_t loci_event_status = CLOCK_XMC1_STATUS_SUCCESS;
  CLOCK_XMC1_STATUS_t stdbyclkfail_status = CLOCK_XMC1_STATUS_SUCCESS;
  CLOCK_XMC1_STATUS_t loss_ext_clock_event_status = CLOCK_XMC1_STATUS_SUCCESS;
  CLOCK_XMC1_STATUS_t dco1_out_sync_status = CLOCK_XMC1_STATUS_SUCCESS;
  if (handle->init_status == false) {
    #ifdef CLOCK_XMC1_INTERRUPT_ENABLED
    status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_Init(handle->global_scu_handlerptr);
    if (CLOCK_XMC1_STATUS_SUCCESS == status) {
      #ifdef CLOCK_XMC1_LOCI_EVENT_ENABLED
      /* Initialization of CPU_CTRL_XMC1 APP */
      loci_event_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(GLOBAL_SCU_XMC1_EVENT_LOCI, handle->callback_function_loci);
      /* Enable Loss of DCO1 Clock Event */
      XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_LOCI);
      #endif
    }  
    #ifdef CLOCK_XMC1_STDBYCLKFAIL_EVENT_ENABLED
    /* Initialization of CPU_CTRL_XMC1 APP */
    stdbyclkfail_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(GLOBAL_SCU_XMC1_EVENT_STDBYCLKFAIL, handle->callback_function_std
/* Enable Standby Clock Failure Event */
XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_STDBYCLKFAIL);
#endif
#if (UC_SERIES == XMC14)
#endif
#endif
/* Initialization of CPU_CTRL_XMC1 APP */
loss_ext_clock_event_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(
GLOBAL_SCU_XMC1_EVENT_LOSS_EXT_CLOCK, handle->callback_function_loss_ext_clock);
/* Enable Loss of external OSC_HP clock Event */
XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_LOSS_EXT_CLOCK);
#endif
#endif
/* Initialization of CPU_CTRL_XMC1 APP */
dco1_out_sync_status = (CLOCK_XMC1_STATUS_t)GLOBAL_SCU_XMC1_RegisterCallback(
GLOBAL_SCU_XMC1_EVENT_DCO1_OUT_SYNC, handle->callback_function_dco1_out_sync);
/* Enable DCO1 Out of SYNC Event */
XMC_SCU_INTERRUPT_EnableEvent(GLOBAL_SCU_XMC1_EVENT_DCO1_OUT_SYNC);
#endif
00133
status = (CLOCK_XMC1_STATUS_t)(((uint32_t)loci_event_status) | ((uint32_t)stdbyclkfail_status) |
																																								((uint32_t)loss_ext_clock_event_status) | ((uint32_t)dco1_out_sync_status));

if (CLOCK_XMC1_STATUS_SUCCESS == status) {
    handle->init_status = true;
}

return (status);

/* API for ramping up/down the system clock frequency */
void CLOCK_XMC1_SetMCLKFrequency(uint32_t freq_khz) {
    XMC_SCU_CLOCK_SetMCLKFrequency(freq_khz);
}

#if (CLOCK_XMC1_OSCHP_ENABLED)
/* API to retrieve high precision external oscillator frequency */
uint32_t OSCHP_GetFrequency(void) {
    return (CLOCK_XMC1_OSCHP_FREQUENCY);
}
#endif
#if (CLOCK_XMC1_DCO1_CALIBRATION_ENABLED)
/* API to check whether DCO1 is synchronize
bool CLOCK_XMC1_IsDCO1ExtRefCalibrationReady(void) {
    return (XMC_SCU_CLOCK_IsDCO1ExtRefCalibrationReady());
}
#endif