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>Definition</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>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>WDT</td>
<td>Watch Dog Timer</td>
</tr>
<tr>
<td>CCU</td>
<td>Capture Compare Unit</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>ETH</td>
<td>Ethernet</td>
</tr>
<tr>
<td>EBU</td>
<td>External Bus Unit</td>
</tr>
<tr>
<td>SCU</td>
<td>System Control Unit</td>
</tr>
<tr>
<td>RTC</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>OFI</td>
<td>Fast Internal Clock,</td>
</tr>
<tr>
<td>OHP</td>
<td>High Precision Crystal Oscillator</td>
</tr>
<tr>
<td>LHP</td>
<td>Low Precision Crystal Oscillator</td>
</tr>
<tr>
<td>ULP</td>
<td>Ultra Low Power Oscillator</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>OSI</td>
<td>Internal Slow Oscillator</td>
</tr>
<tr>
<td>STDBY</td>
<td>Standby</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Secure Digital / Multi Media Card (Interface)</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>PB</td>
<td>Peripheral Bridge</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input Output</td>
</tr>
<tr>
<td>PERI</td>
<td>Peripheral</td>
</tr>
<tr>
<td>NMI</td>
<td>Non Maskable Interrupt</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage Controlled Oscillator</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</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_XMC4

Overview

The **CLOCK_XMC4** APP provides the following functionalities:

1. Setting the sources of various clocks in xmc4 devices.
2. Calibration selection for internal fast oscillator settings.
3. Configuration of Main / System PLL settings.
4. Conditional configuration of external clock output settings, to monitor a various clocks: System Clock, Main PLL clock, USB PLL clock, Standby clock (only for xmc42/41 devices).
5. Setting the entire clock setup using `SystemCoreClockSetup()`.
6. Runtime clock changes are supported.

Note:

1. By default, the SystemInit() calls weak API `SystemCoreClockSetup()` which exits in system_XMC4x.c file. When the **CLOCK_XMC4** APP used in the project, then SystemInit() calls the SystemCoreClockSetup() which exists in clock_xmc4_conf.c file.
Figure 1: Hardware and Software connectivity of CLOCK_XMC4 APP

Figure 1, shows how the APP is structured in DAVE™. The CLOCK_XMC4 APP uses SCU module to generate a various clocks such as: System clock, peripheral clock backup clock, Standby clock etc. And it also uses GPIO module to monitor an external clock output.
Figure 2 shows how the various clocks are derived from the source.

Note:

1. For XMC42/1 devices, the maximum PLL clock and system clock frequency are limited to 80MHz.
2. The clock generated for the various unit of device may differ based on other XMC4x derivatives.
3. Additionally the APP is checking for appropriate clock ratio combinations between fCCU, fCPU and fPERIPH. The valid values of clock divide registers for fCCU, fCPU and fPERIPH clocks are mentioned below.
Figure 3: Valid clock ratio combinations between fCCU, fCPU and fPERIPH

Supported Devices

The APP supports below devices:

1. XMC4800 Series
2. XMC4700 Series
3. XMC4500 Series
4. XMC4400 Series
5. XMC4300 Series
6. XMC4200 / XMC4100 Series

References

1. XMC4800 Reference Manual
2. XMC4700 Reference Manual
5. XMC4300 Reference Manual
7. XMC4100 Reference Manual

Limitations

None
The above figure 1 represents the internal software architecture of the CLOCK_XMC4 APP. A CLOCK_XMC4 APP instance exists in a
DAVE™ project with fixed attributes as shown. The APP configures SCU clock module, and conditionally GPIO module to monitor a various clocks. This in addition requires the consumption of the CPU_CTRL_XMC4 APP for handling the NMI trap based on trap event selection in event settings page of CLOCK_XMC4 APP GUI.

**CLOCK_XMC4** is used by use-case APPs like: UART, PWM, SPI, CAN (top level) APPs.

**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_ccu_output</td>
<td>Output</td>
<td>Always</td>
<td>Clock input to CCU4, CCU8 and POSIF modules</td>
</tr>
<tr>
<td>clk_perbridge_output</td>
<td>Output</td>
<td>Always</td>
<td>Clock input to DSD module</td>
</tr>
</tbody>
</table>

Figure 2 explains the preferred way of clock initialization sequence, and is being used in SCU low-level driver.
Figure 2: Clock Initialization Sequence
Figure 1: Clock Control Settings
<table>
<thead>
<tr>
<th>Clock Control Settings</th>
<th>Clock Generation Settings</th>
<th>Standby Clock Generation Settings</th>
<th>Clock Selection Settings</th>
<th>Event Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High Precision Oscillator Settings</strong>&lt;br&gt;Operating mode:</td>
<td>External Clock Input Mode</td>
<td>External Clock Input Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External clock frequency [MHz]:</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Internal Fast Oscillator Settings</strong>&lt;br&gt;Calibration:</td>
<td>Factory Calibration</td>
<td>Factory Calibration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Main PLL Settings</strong>&lt;br&gt;Enable main PLL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLL clock source:</td>
<td>External Crystal High Precision Oscillator</td>
<td>External Crystal High Precision Oscillator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLL operating mode:</td>
<td>Normal Mode</td>
<td>Normal Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requested PLL frequency [MHz]:</td>
<td>288</td>
<td>Actual PLL frequency [MHz]:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>USB PLL Settings</strong>&lt;br&gt;Enable USB PLL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requested USB PLL frequency [MHz]:</td>
<td></td>
<td>Actual USB PLL frequency [MHz]:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2: Clock Generation Settings**
Figure 3: Standby Clock Generation Settings

<table>
<thead>
<tr>
<th>Standby Clock (IFSTDBY)</th>
<th>RTC Clock (RTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock source:</strong></td>
<td><strong>Clock source:</strong></td>
</tr>
<tr>
<td>Internal Slow Oscillator</td>
<td>Internal Slow Oscillator</td>
</tr>
<tr>
<td>Clock Control Settings</td>
<td>Clock Generation Settings</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>System Clock (SYS)</strong></td>
<td></td>
</tr>
<tr>
<td>Clock source:</td>
<td>Main PLL Clock</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>2</td>
</tr>
<tr>
<td><strong>CPU Clock (CPU)</strong></td>
<td></td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>Peripheral Bus Clock (PERIPH)</strong></td>
<td></td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>CCU Clock (CCU)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable CCU clock</td>
<td></td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>USB Clock (USB) and SDMMC Clock (SDMMC)</strong></td>
<td></td>
</tr>
<tr>
<td>Checkmark</td>
<td>Enable USB clock</td>
</tr>
<tr>
<td>Clock source:</td>
<td>Main PLL Clock</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>6</td>
</tr>
<tr>
<td><strong>Watchdog Clock (WDWT)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable Watchdog clock</td>
<td></td>
</tr>
<tr>
<td>Clock source:</td>
<td>Internal Fast Oscillator</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>EBU Clock (EBU)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable EBU clock</td>
<td></td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>External Clock (EXT)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable external clock output</td>
<td></td>
</tr>
<tr>
<td>Clock source:</td>
<td>System Clock</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>1</td>
</tr>
<tr>
<td><strong>Ethernet Clock (ETH)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable Ethernet clock</td>
<td></td>
</tr>
<tr>
<td>Clock source:</td>
<td>System Clock</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>2</td>
</tr>
<tr>
<td><strong>EtherCAT Clock (ECAT)</strong></td>
<td>Checkmark</td>
</tr>
<tr>
<td>Enable EtherCAT clock</td>
<td></td>
</tr>
<tr>
<td>Clock source:</td>
<td>USB PLL Clock</td>
</tr>
<tr>
<td>Clock divider:</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 4: Clock Selection Settings

<table>
<thead>
<tr>
<th>Clock Control Settings</th>
<th>Clock Generation Settings</th>
<th>Standby Clock Generation Settings</th>
<th>Clock Selection Settings</th>
<th>Event Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ OSC_HP oscillator watchdog trap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ USB VCO lock trap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ System VCO lock trap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ OSC_ULP oscillator watchdog trap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Clock events are handled in user defined NMI_Handler function.

Figure 5: Event Settings
CLOCK_XMC4

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

## Enumerations

```c
enum CLOCK_XMC4_STATUS {
    CLOCK_XMC4_STATUS_SUCCESS = 0U,
    CLOCK_XMC4_STATUS_FAILURE = 1U
}
```
### Enumeration Type Documentation

```c
enum CLOCK_XMC4_STATUS
```

**Enumerator:**

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CLOCK_XMC4_STATUS_SUCCESS</code></td>
<td>APP initialization is success</td>
</tr>
<tr>
<td><code>CLOCK_XMC4_STATUS_FAILURE</code></td>
<td>APP initialization is failure</td>
</tr>
</tbody>
</table>

Definition at line 101 of file `CLOCK_XMC4.h`. 
CLOCK_XMC4

<table>
<thead>
<tr>
<th>Home</th>
<th>Data Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data structures</td>
<td>Data Structures</td>
</tr>
</tbody>
</table>


## Data Structures

<table>
<thead>
<tr>
<th><code>struct</code></th>
<th><code>CLOCK_XMC4</code></th>
<th>Configuration structure for <code>CLOCK_XMC4</code> APP. More...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>typedef struct</code></td>
<td><code>CLOCK_XMC4</code></td>
<td><code>CLOCK_XMC4_t</code></td>
</tr>
</tbody>
</table>
# CLOCK_XMC4

## Methods

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC4_GetAppVersion (void)</td>
<td>Get CLOCK_XMC4 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC4_STATUS_t</td>
<td>CLOCK_XMC4_Init (CLOCK_XMC4_t *handle)</td>
<td>Initializes a CLOCK_XMC4 APP instance.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>OSCHP_GetFrequency (void)</td>
<td>This is a non-weak function, which retrieves high precision external oscillator frequency. Note: This function is used by xmc4_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC4_StepSystemPllFrequency (uint32_t kdiv)</td>
<td>API for ramping down the system PLL clock frequency.</td>
</tr>
</tbody>
</table>
Function Documentation

DAVE_APP_VERSION_t CLOCK_XMC4_GetAppVersion (void)

Get CLOCK_XMC4 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_XMC4 APP:
    // SystemCoreClockSetup() is called from SystemInit().
    init_status = DAVE_Init();

    version = CLOCK_XMC4_GetAppVersion();
    if (version.major != 1U) {
        // Probably, not the right version.
    }

    // More code here
    while(1) {
    }
}
```
Definition at line 83 of file CLOCK_XMC4.c.

CLOCK_XMC4_STATUS_t CLOCK_XMC4_Init (CLOCK_XMC4_t * h)

Initializes a CLOCK_XMC4 APP instance.

**Parameters:**
- `handle` address of CLOCK_XMC4 APP handler

**Returns:**
- CLOCK_XMC4_STATUS_SUCCESS : if initialization is successful
- CLOCK_XMC4_STATUS_FAILURE : if initialization is failed

**Description:**
CLOCK_XMC4_Init API is called during initialization of DAVE APPS. This API Initializes NMI TRAP Configuration.

**Example Usage:**

```c
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t status;

    status = DAVE_Init(); // CLOCK_XMC4_Init API is called during initialization of DAVE APPS
    if(DAVE_STATUS_SUCCESS == status)
    {
        // user code
    }
}
while(1) {
    }
} return (1);
}

Definition at line 96 of file CLOCK_XMC4.c.

References CLOCK_XMC4_STATUS_SUCCESS, and CLOCK_XMC4::init_status.

**void CLOCK_XMC4_StepSystemPllFrequency (uint32_t kdiv)**

API for ramping down the system PLL clock frequency.

**Parameters:**
- **kdiv** PLL output divider K2DIV. Range: 1 to 128. Represents (K2DIV+1).

**Returns:**
- none

**Description:**
The function can be used for ramping down the system PLL clock frequency.

**Example Usage:**

```c
#include <DAVE.h>

int main(void) {
```
DAVE_STATUS_t init_status;
uint32_t kdiv = 10U; // (K2DIV+1) value for scaling down the system PLL clock frequency

// Initialize CLOCK_XMC4 APP:
// SystemCoreClockSetup() is called from SystemInit().
init_status = DAVE_Init();
if (DAVE_STATUS_SUCCESS == init_status)
{
    // More code here

    // User decided to reduce the system power consumption by scaling down the system PLL clock frequency
    CLOCK_XMC4_StepSystemPllFrequency(kdiv); // PLL frequency is scaling down by K2DIV factor.

    // More code here
    while(1) {
    }
}
return (1);

Definition at line 116 of file CLOCK_XMC4.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 xmc4_scu LLD for internal
operations. Therefore the user do not required to call this API explicitly.

Returns:
   uint32_t Range: 4 to 25 in External Crystal Mode, 4 to 40 in 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 109 of file CLOCK_XMC4.c.
**CLOCK_XMC4**

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

**Usage**

*Below shows typical usages of CLOCK_XMC4 APP.*

**Use case 1:**
This example monitors the system clock frequency via EXTCLK pin.

**Instantiate the required APPs**
Drag an instance of CLOCK_XMC4. Update the fields in the GUI of this APP with the following configuration.

**Configure the APP**
CLOCK_XMC4:
<table>
<thead>
<tr>
<th>Clock Control Settings</th>
<th>Clock Generation Settings</th>
<th>Standby Clock Generation Settings</th>
<th>Clock Selection Settings</th>
<th>Event Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Clock (S)YS</strong></td>
<td><strong>Main PLL Clock</strong></td>
<td><strong>4</strong> Actual frequency [MHz]: 72</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CPU Clock (fCPU)</strong></td>
<td></td>
<td><strong>1</strong> Actual frequency [MHz]: 72</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Peripheral Bus Clock (IPERIP)</strong></td>
<td></td>
<td><strong>1</strong> Actual frequency [MHz]: 72</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CCU Clock (ICCU)</strong></td>
<td><strong>Enable CCU clock</strong></td>
<td><strong>1</strong> Actual frequency [MHz]: 72</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>USB Clock (USB) and SDMMC Clock (ISDMMCI)</strong></td>
<td><strong>Enable USB / SDMMC clock</strong></td>
<td><strong>1</strong> Actual frequency [MHz]: 40</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Watchdog Clock (fWDT)</strong></td>
<td><strong>Enable Watchdog clock</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EBU Clock (fEBU)</strong></td>
<td><strong>Enable EBU clock</strong></td>
<td><strong>1</strong> Actual frequency [MHz]: 288</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>External Clock (fEXT)</strong></td>
<td><strong>Enable external clock output</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ethernet Clock (fETH)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EtherCAT Clock (fECAT)</strong></td>
<td><strong>Enable EtherCAT clock</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clock sources</strong></td>
<td><strong>Clock divider</strong></td>
<td><strong>Actual frequency [MHz]</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Configure System clock divider as 4
2. Select external clock output and System Clock as external clock output source

Manual pin allocation

![Manual Pin Allocator]

1. Select a pin to monitor System clock frequency
   **Note:** The pin number is specific to the development board chosen to run this example. The pin shown in the image above may not be available on every XMC boot kit. Ensure that a proper pin is selected according to the board.

Generate code
Files are generated here: `<project_name>/Dave/Generated/`
(`project_name` is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.
**Note:** Code must be explicitly generated for every change in the GUI configuration.
**Important:** Any manual modification to APP specific files will be overwritten by a subsequent code generation operation.

Build and Run the Project
Observation
The configured system clock frequency (72MHz for XMC48/47/43 devices, 30MHz for XMC45/44 devices, 20MHz for XMC42 devices) must observe in the oscilloscope at pin P1.15.

Use case 2:
This example enables user to monitor OSC_HP frequency (fOSC) whether it is usable for the VCO as a part of the PLL or not. And also allow user to take appropriate action when NMI TRAP occurs.

Instantiate the required APPs
Drag an instance of CLOCK_XMC4 APP and CPU_CTRL_XMC4 APP. Update the fields in the GUI of this APP with the following configuration.

Configure the APP
CLOCK_XMC4:

1. Enable OSC_HP oscillator watchdog trap
2. Enable USB VCO lock trap
3. Enable System VCO lock trap
4. Enable OSC_ULP oscillator watchdog trap

Note: User must define the NMI TRAP handler as void NMI_Handler(void) for handling trap.

Generate code
Files are generated here: `<project_name>/Dave/Generated/'
('project_name' is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.

**Note:** Code must be explicitly generated for every change in the GUI configuration.

**Important:** Any manual modification to APP specific files will be overwritten by a subsequent code generation operation.

**Sample Application (main.c)**

```c
#include <DAVE.h>    //Declarations from DAVE Code Generation (includes SFR declaration)

volatile uint32_t TRAP_OSC_WDG_FLAG=0;
volatile uint32_t TRAP_VCO_LOCK_FLAG=0;
volatile uint32_t TRAP_USB_VCO_FLAG=0;
volatile uint32_t TRAP_ULP_WDG_FLAG=0;

void NMI_Handler(void)
{
    uint32_t TRAP_FLAG=0;
    TRAP_FLAG = XMC_SCU_TRAP_GetStatus();
    if((TRAP_FLAG & XMC_SCU_TRAP_OSC_WDG) >> SCU_TRAP_TRAPSTAT_SOSCWWDGT_Pos)
    {
        TRAP_OSC_WDG_FLAG++;
        XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_OSC_WDG);
        // Add application code here for handling OSC_HP oscillator watchdog trap
    }

    if((TRAP_FLAG & XMC_SCU_TRAP_VCO_LOCK) >> SCU_TRAP_TRAPSTAT_SVCOLCKT_Pos)
    {
        TRAP_VCO_LOCK_FLAG++;
    }
```

XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_VCO_LOCK);
    // Add application code here for handling system VCO lock trap
}
if((TRAP_FLAG & XMC_SCU_TRAP_USB_VCO_LOCK) >> SCU_TRAP_TRAPSTAT_UVCOLCKT_Pos)
{
    TRAP_USB_VCO_FLAG++;
    XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_USB_VCO_LOCK);
    // Add application code here for handling USB VCO lock trap
}
if((TRAP_FLAG & XMC_SCU_TRAP_ULP_WDG) >> SCU_TRAP_TRAPSTAT_ULPWGDGT_Pos)
{
    TRAP_ULP_WDG_FLAG++;
    XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_ULP_WDG);
    // Add application code here for handling OSC_ULP oscillator watchdog trap
}
}

// @brief main() - Application entry point
//
// <b>Details of function</b><br>
// This routine is the application entry point. It is invoked by the device startup code. It is responsible for
// invoking the App initialization dispatcher routine - DAVE_Init() and hosting the place-holder for user application
// code.
//
int main(void)
{ 
    DAVE_STATUS_t status;
    status = DAVE_Init(); // CLOCK_XMC4_Init() is called from within DAVE_Init().
    if(status == DAVE_STATUS_FAILURE) 
    { 
        // Placeholder for error handler code. The while loop below can be replaced with an user error handler
        XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
        while(1U)
        {
        }
    }
    // Placeholder for user application code. The while loop below can be replaced with user application code.
    while(1U)
    {
    }
    return (1);
}

Build and Run the Project
Note: User must add application code for ensuring the safety operation.

Observation
TRAP_OSC_WDG_FLAG, TRAP_VCO_LOCK_FLAG, TRAP_USB_VCO_FLAG and TRAP_ULP_WDG_FLAG flags can be monitored for ensuring the safety operation.
CLOCK_XMC4

Release History

Release History
## Data Structures

Here are the data structures with brief descriptions:

| CLOCK_XMC4 | Configuration structure for CLOCK_XMC4 APP |
# CLOCK_XMC4

[CLOCK_XMC4 Struct Reference](data-structures)

<table>
<thead>
<tr>
<th>Data Structures</th>
<th>Data Structure Index</th>
<th>Data Fields</th>
</tr>
</thead>
</table>

CLOCK_XMC4 Struct

Data structures
Detailed Description

Configuration structure for CLOCK_XMC4 APP.

Definition at line 121 of file CLOCK_XMC4.h.

#include <CLOCK_XMC4.h>
## Data Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>init_status</td>
</tr>
</tbody>
</table>
**Field Documentation**

<table>
<thead>
<tr>
<th>bool CLOCK_XMC4::init_status</th>
</tr>
</thead>
</table>

APP is initialized or not.

Definition at line 123 of file CLOCK_XMC4.h.

Referenced by CLOCK_XMC4_Init().

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

- CLOCK_XMC4.h
## CLOCK_XMC4

<table>
<thead>
<tr>
<th>Data Structures</th>
<th>Data Structure Index</th>
<th>Data Fields</th>
</tr>
</thead>
</table>

### Data Structure Index

<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CLOCK_XMC4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
</table>
Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- `init_status : CLOCK_XMC4`
CLOCK_XMC4

<table>
<thead>
<tr>
<th>Home</th>
<th>Data Structures</th>
<th>Data Structure Index</th>
<th>Data Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Variables</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- init_status: CLOCK_XMC4
CLOCK_XMC4

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

- CLOCK_XMC4.c [code]
- CLOCK_XMC4.h [code]
## CLOCK_XMC4

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

**CLOCK_XMC4.c File Reference**
Detailed Description

Date:
   2016-07-08

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_XMC4.c.

#include "clock_xmc4.h"
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC4_GetAppVersion (void)</td>
<td>Get CLOCK_XMC4 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC4_STATUS_t</td>
<td>CLOCK_XMC4_Init (CLOCK_XMC4_t *handle)</td>
<td>Initializes a CLOCK_XMC4 APP instance.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>OSCHP_GetFrequency (void)</td>
<td>This is a non-weak function, which retrieves high precision external oscillator frequency.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: This function is used by xmc4_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC4_StepSystemPllFrequency (uint32_t kdiv)</td>
<td>API for ramping down the system PLL clock frequency.</td>
</tr>
</tbody>
</table>
Function Documentation

CLOCK_XMC4_STATUS_t CLOCK_XMC4_Init (CLOCK_XMC4_t * h)

Initializes a CLOCK_XMC4 APP instance.

Parameters:

handle address of CLOCK_XMC4 APP handler

Returns:

CLOCK_XMC4_STATUS_SUCCESS : if initialization is successful
CLOCK_XMC4_STATUS_FAILURE : if initialization is failed

Description:

CLOCK_XMC4_Init API is called during initialization of DAVE APPS. This API initializes NMI TRAP Configuration.

Example Usage:

```c
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t status;

    status = DAVE_Init(); // CLOCK_XMC4_Init API is called during initialization of DAVE APPS
    if(DAVE_STATUS_SUCCESS == status)
    {
        // user code

        while(1)
        {
        }
    }
```
Definition at line 96 of file CLOCK_XMC4.c.

References CLOCK_XMC4_STATUS_SUCCESS, and CLOCK_XMC4::init_status.

```c
void CLOCK_XMC4_StepSystemPllFrequency ( uint32_t kdiv )
```

API for ramping down the system PLL clock frequency.

**Parameters:**
- `kdiv` PLL output divider K2DIV. Range: 1 to 128. Represents (K2DIV+1).

**Returns:**
- none

**Description:**
The function can be used for ramping down the system PLL clock frequency.

**Example Usage:**
```c
#include <DAVE.h>

int main(void)
{
    DAVE_STATUS_t init_status;
    uint32_t kdiv = 10U;  // (K2DIV+1) value for scaling down the system PLL clock frequency
    // Initialize CLOCK_XMC4 APP:
    // SystemCoreClockSetup() is called from Syste
mInit().
    init_status = DAVE_Init();
    if(DAVE_STATUS_SUCCESS == init_status)
    {
        // More code here
    
        // User decided to reduce the system power consumption by scaling down the system PLL clock frequency
        CLOCK_XMC4_SeroptSystemPllFrequency(kdiv); // PLL frequency is scaling down by K2DIV factor.
    
        // More code here
        while(1) {

            }
        }
        return (1);
    }

Definition at line 116 of file CLOCK_XMC4.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 xmc4_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.

Returns:
    uint32_t Range: 4 to 25 in External Crystal Mode, 4 to 40 in
External External Direct Input Mode.

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

Definition at line 109 of file `CLOCK_XMC4.c`.

Go to the source code of this file.
CLOCK_XMC4

<table>
<thead>
<tr>
<th>Home</th>
<th>File List</th>
<th>Globals</th>
<th>Data Structures</th>
</tr>
</thead>
</table>

CLOCK_XMC4.h File Reference
**Detailed Description**

**Date:**

2016-07-08

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_XMC4.h**.

```c
#include <xmc_scu.h> #include <xmc_gpio.h>
#include <DAVE_Common.h>
#include "clock_xmc4_conf.h"
#include "clock_xmc4Extern.h"
```
## Data Structures

```c
struct CLOCK_XMC4

Configuration structure for CLOCK_XMC4 APP. More...
```
typedef struct CLOCK_XMC4 CLOCK_XMC4_t
Configuration structure for CLOCK_XMC4 APP.
## Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVE_APP_VERSION_t</td>
<td>CLOCK_XMC4_GetAppVersion (void)</td>
<td>Get CLOCK_XMC4 APP version.</td>
</tr>
<tr>
<td>CLOCK_XMC4_STATUS_t</td>
<td>CLOCK_XMC4_Init (CLOCK_XMC4_t *handle)</td>
<td>Initializes a CLOCK_XMC4 APP instance.</td>
</tr>
<tr>
<td>uint32_t</td>
<td>OSCHP_GetFrequency (void)</td>
<td>This is a non-weak function, which retrieves high precision external oscillator frequency.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This function is used by xmc4_scu LLD for internal operations. Therefore the user do not required to call this API explicitly.</td>
</tr>
<tr>
<td>void</td>
<td>CLOCK_XMC4_StepSystemPllFrequency (uint32_t kdiv)</td>
<td>API for ramping down the system PLL clock frequency.</td>
</tr>
<tr>
<td>enum</td>
<td>CLOCK_XMC4_STATUS {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLOCK_XMC4_STATUS_SUCCESS = 0U</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLOCK_XMC4_STATUS_FAILURE = 1U</td>
<td></td>
</tr>
</tbody>
</table>

Go to the source code of this file.
CLOCK_XMC4

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

- `CLOCK_XMC4_GetAppVersion()` : CLOCK_XMC4.c, CLOCK_XMC4.h
- `CLOCK_XMC4_Init()` : CLOCK_XMC4.h, CLOCK_XMC4.c
- `CLOCK_XMC4_STATUS` : CLOCK_XMC4.h
- `CLOCK_XMC4_STATUS_FAILURE` : CLOCK_XMC4.h
- `CLOCK_XMC4_STATUS_SUCCESS` : CLOCK_XMC4.h
- `CLOCK_XMC4_StepSystemPllFrequency()` : CLOCK_XMC4.c, CLOCK_XMC4.h
- `CLOCK_XMC4_t` : CLOCK_XMC4.h
- `OSCHP_GetFrequency()` : CLOCK_XMC4.c, CLOCK_XMC4.h
CLOCK_XMC4

- CLOCK_XMC4_GetAppVersion() : CLOCK_XMC4.c , CLOCK_XMC4.h
- CLOCK_XMC4_Init() : CLOCK_XMC4.h , CLOCK_XMC4.c
- CLOCK_XMC4_StepSystemPllFrequency() : CLOCK_XMC4.h , CLOCK_XMC4.c
- OSCHP_GetFrequency() : CLOCK_XMC4.c , CLOCK_XMC4.h
### CLOCK_XMC4

- **CLOCK_XMC4_t**: `CLOCK_XMC4.h`

```c
// Clock_XMC4_t definition
```
```c
// Define CLOCK_XMC4_t
```
• CLOCK_XMC4_STATUS : CLOCK_XMC4.h
CLOCK_XMC4

- CLOCK_XMC4_STATUS_FAILURE : CLOCK_XMC4.h
- CLOCK_XMC4_STATUS_SUCCESS : CLOCK_XMC4.h
#ifndef CLOCK_XMC4_H
#define CLOCK_XMC4_H

#include <xmc_scu.h>
#include <xmc_gpio.h>
#include <DAVE_Common.h>
#include "clock_xmc4_conf.h"

/* HEADER FILES */

/* MACROS */
#if (!((XMC_LIB_MAJOR_VERSION == 2U) && \
  (XMC_LIB_MINOR_VERSION >= 0U) && \
  (XMC_LIB_PATCH_VERSION >= 0U)))
#error "CLOCK_XMC4 requires XMC Peripheral Library v2.0.0 or higher"
#endif

/****************************
* ENUMS
**************************
/*
@brief enumeration for CLOCK_XMC4 APP */
typedef enum CLOCK_XMC4_STATUS
{
  CLOCK_XMC4_STATUS_SUCCESS = 0U,
  CLOCK_XMC4_STATUS_FAILURE = 1U
} CLOCK_XMC4_STATUS_t;

/****************************
* DATA STRUCTURES
*************************/
typedef struct CLOCK_XMC4
{
  bool init_status;
} CLOCK_XMC4_t;
#endif __cplusplus

extern "C" {
# API Prototypes

```c
/*
Include APP extern declaration file */
#include"clock_xmc4Extern.h"
```

```c
DAVE_APP_VERSION_t CLOCK_XMC4_GetAppVersion(void);

CLOCK_XMC4_STATUS_t CLOCK_XMC4_Init(CLOCK_XMC4_t *handle);

#ifdef CLOCK_XMC4_OSCHP_ENABLED
uint32_t OSCHP_GetFrequency(void);
#endif

void CLOCK_XMC4_StepSystemPllFrequency(uint32_t kdiv);

#ifdef __cplusplus
}
#endif
```

```c
/* End of CLOCK_XMC4_H */
```
CLOCK_XMC4

Go to the documentation of this file.

```c
#include "clock_xmc4.h"
```
/**************************************************
************ LOCAL ROUTINES
**************************************************

/* API to retrieve version of the APP */
DAVE_APP_VERSION_t CLOCK_XMC4_GetAppVersion(void)
{
    DAVE_APP_VERSION_t version;
    version.major = (uint8_t)CLOCK_XMC4_MAJOR_VERSION;
    version.minor = (uint8_t)CLOCK_XMC4_MINOR_VERSION;
    version.patch = (uint8_t)CLOCK_XMC4_PATCH_VERSION;
    return (version);
}

/* API to initialize the CLOCK_XMC4 APP TRAP events */
CLOCK_XMC4_STATUS_t CLOCK_XMC4_Init(CLOCK_XMC4_t *handle)
CLOCK_XMC4_STATUS_t status = CLOCK_XMC4_STATUS_SUCCESS;

XMC_ASSERT("CLOCK_XMC4 APP handle function pointer uninitialized", (handle != NULL));

handle->init_status = true;

return (status);

#ifdef CLOCK_XMC4_OSCHP_ENABLED

/* API to retrieve high precision external oscillator frequency */

uint32_t OSCHP_GetFrequency(void)
{
    return (CLOCK_XMC4_OSCHP_FREQUENCY);
}
#endif

/* API for ramping down the system PLL clock frequency */

void CLOCK_XMC4_StepSystemPllFrequency(uint32_t kdiv)
{
    XMC_ASSERT("Incorrect kdiv value", ((kdiv >= 1) && (kdiv >= 128)));
    XMC_SCU_CLOCK_StepSystemPllFrequency(kdiv);
}