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

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).

Home

Abbreviations and Definitions

Abbreviations and Definitions

Abbrevia	tions:
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
10	Input Output
PLL	Phase Locked Loop
WDT	Watch Dog Timer
CCU	Capture Compare Unit
USB	Universal Serial Bus
CPU	Central Processing Unit
ETH	Ethernet
EBU	External Bus Unit
SCU	System Control Unit
RTC	Real Time Clock
OFI	Fast Internal Clock,
OHP	High Precision Crystal Oscillator
LHP	Low Precision Crystal Oscillator

ULP	Ultra Low Power Oscillator
OSI	Internal Slow Oscillator
STDBY	Standby
SDMMC	Secure Digital / Multi Media Card (Interface)
DMA	Direct Memory Access
PB	Peripheral Bridge
GPIO	General Purpose Input Output
PERI	Peripheral
NMI	Non Maskable Interrupt
VCO	Voltage Controlled Oscillator
UART	Universal Asynchronous Receiver Transmitter
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
CAN	Controller Area Network
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

Home

Overview

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:

 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 : Clock selection & generation unit

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

CCUCLKCR.CCUDIV + 1	CPUCLKCR.CPUDIV + 1	PBCLKCR.PBDIV + 1
1	1	1
1	1	2
1	2	1
2	1	2
2	2	1

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
- 3. XMC4500 Reference Manual
- 4. XMC4400 Reference Manual
- 5. XMC4300 Reference Manual
- 6. XMC4200 Reference Manual
- 7. XMC4100 Reference Manual

Limitations

None

Home

Architecture Description

Architecture Description



Figure 1 : Architecture of CLOCK_XMC4 APP

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

Signal Name	Input/Output	Availability	Description
clk_ccu_output	Output	Always	Clock input to CCU4, CCU8 and POSIF modules
clk_perbridge_output	Output	Always	Clock input to DSD module

Figure 2 explains the preferred way of clock initialization sequence, and is being used in SCU low-level driver.



Figure 2 : Clock Initialization Sequence

Home

APP Configuration Parameters

App Configuration Parameters



Figure 1: Clock Control Settings

Operating mode: E	xternal Clock Input Mode 👻
External clock frequency [MHz]: 1	.2
Internal Fast Oscillator Settings	
Calibration: F	actory Calibration
Main PLL Settings	
M Enable main PLL	
PLL clock source:	External Crystal High Precision Oscillator 🔻
PLL operating mode:	Normal Mode 👻
Requested PLL frequency [MHz]:	288 Actual PLL frequency [MHz]: 288
USB PLL Settings	
Enable USB PLL	
	200 Actual USB PLL frequency [MHz]: 200
Requested USB PLL frequency [MHz]:	

Figure 2: Clock Generation Settings



Clock Control Se	ttings	Clock Generation Settings	Standby Clock Generation Settings	Clock Selection Settings	Event Settings
Standby Clock	(fSTDB	Y)			
Clock source:	Intern	al Slow Oscillator	-		
DTC CL - L ((D)					
- KTC Clock (TK	()				
Clock source:	Intern	al Slow Oscillator	-		
	-				

Figure 3: Standby Clock Generation Settings



	set in set ings
System Clock (fSYS)	
Clock source:	Main PLL Clock 👻
Clock divider:	2 Actual frequency [MHz]: 144
CPU Clock (fCPU)	
Clock divider:	1 Actual frequency [MHz]: 144
Peripheral Bus Clock	(fPERIPH)
Clock divider:	1 Actual frequency [MHz]: 144
CCU Clock (fCCU)	
Clock dividen	1 Actual fragmency (MdLa), 144
Clock divider:	Actual frequency [Minz]: 144
USB Clock (fUSB) and	SDMMC Clock (fSDMMC)
Clock sources	Main PLI Clock
Clock dividen	6 Actual frequency [MHz]: 48
CIOCK divider:	Actual frequency [MIN2]: 40
Watchdog Clock (fW	DT)
	Internal Fact Occillator
Clock source:	
Clock divider:	Actual frequency [MHZ]: 24
EBU Clock (fEBU)	
Enable EBU clock	
Clock divider:	1 Actual frequency [MHz]: 288
External Clock (fEXT)	
	System Clock
Clock source:	System clock
Clock divider:	I Actual frequency [MHz]: 144
a de la competition d	
Ethernet Clock (fETH)) ock
Ethernet Clock (fETH) Enable Ethernet clo Clock source:) ock System Clock +
Ethernet Clock (fETH) Enable Ethernet clo Clock source: Clock divider:) ock System Clock 2 Actual frequency [MHz]: 72
Ethernet Clock (fETH) Enable Ethernet clo Clock source: Clock divider: EtherCAT Clock (fEC/) pock System Clock 2 Actual frequency [MHz]: 72 AD
Ethernet Clock (fETH) Enable Ethernet clo Clock source: Clock divider: EtherCAT Clock (fECA EtherCAT clock (fECA) pock System Clock 2 Actual frequency [MHz]: 72 AT) lock
Ethernet Clock (fETH) Enable Ethernet clo Clock source: Clock divider: EtherCAT Clock (fECA Enable EtherCAT clock (fECA Clock source:) ock System Clock 2 Actual frequency [MHz]: 72 AT) lock USB PLL Clock

Figure 4: Clock Selection Settings

Clock Control Settings Clock Generation Settings Standby Clock Generation Settings Clock Selection Settings Event Settings

OSC_HP oscillator watchdog trap

USB VCO lock trap

System VCO lock trap

OSC_ULP oscillator watchdog trap

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

Figure 5: Event Settings

Home

Enumerations

enum CLOCK_XMC4_STATUS { enum CLOCK_XMC4_STATUS_SUCCESS = 0U, CLOCK_XMC4_STATUS_FAILURE = 1U }

Enumeration Type Documentation

enum CLOCK_XMC4_STATUS

 Enumerator:

 CLOCK_XMC4_STATUS_SUCCESS
 APP initialization is success

 CLOCK_XMC4_STATUS_FAILURE
 APP initialization is failure

 Definition at line 101 of file CLOCK_XMC4.h.

Home

Data Structures

Data structures

Data Structures

struct CLOCK_XMC4 Configuration structure for CLOCK_XMC4 APP. More...

typedef struct CLOCK_XMC4 CLOCK_XMC4_t Configuration structure for CLOCK_XMC4 APP.

Home

Methods

DAVE_APP_VERSION_t	CLOCK_XMC4_GetAppVersion (void) Get CLOCK_XMC4 APP version.
CLOCK_XMC4_STATUS_t	CLOCK_XMC4_Init (CLOCK_XMC4_t *handle) Initializes a CLOCK_XMC4 APP instance.
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.
void	CLOCK_XMC4_StepSystemPllFrequency (uint32_t kdiv) API for ramping down the system PLL clock frequency.

Methods

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 Syste
mInit().
   init_status = DAVE_Init();
   version = CLOCK_XMC4_GetAppVersion();
   if (version.major != 1U) {
     // Probably, not the right version.
   }
   // More code here
  while(1) {
   }
}
```

```
return (0);
```

}

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:

```
#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:

#include <DAVE.h>

```
int main(void)
```

{

```
DAVE_STATUS_t init_status;
   uint32_t kdiv = 10U; // (K2DIV+1) value for s
caling 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 co
nsumption by scaling down the system PLL clock fr
equency
    CLOCK_XMC4_StepSystemPllFrequency(kdiv); // f
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 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**.

Home

Usage

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:

lock Control Settings	Clock Generation Settings Standby Clock Generation Setti	ngs Clock Selection Settings	Event Settings
System Clock (fSYS)			
Clock source:	Main PLL Clock 👻		
Clock divider: 1	4 Actual frequency [MHz]: 72		
CPU Clock (fCPU)			
Clock divider:	1 Actual frequency [MHz]: 72		
Peripheral Bus Clock ((PERIPH)		
Clock divider:	1 Actual frequency [MHz]: 72		
CCU Clock (fCCU)			
Enable CCU clock			
Clock divider:	1 Actual frequency [MHz]: 72		
USB Clock (fUSB) and	SDMMC Clock (fSDMMC)		
🖉 Enable USB / SDMN	1C clock		
Clock source:	Main PLL Clock 👻		
Clock divider:	6 Actual frequency [MHz]: 48		
	T)		
Enable Watchdog c	lock		
Clock source:	Internal Fast Oscillator 👻		
Clock divider:	1 Actual frequency [MHz]: 24		
FRU Clock (FFRU)			
Enable EBU clock			
Clock divider:	1 Actual frequency [MHz]: 288		
External Clock (fEXT) –			
Enable external close	:k output ②		
Clock source:	System Clock 👻		
Clock divider:	1 Actual frequency [MHz]: 72		
Ethernet Clock (fFTH)			
Enable Ethernet clo	ck		
Clock source:	System Clock 👻		
Clock divider:	2 Actual frequency [MHz]: 36		
EtherCAT Clock (fECA	T)		
Enable EtherCAT cl	ock		
Clock source:	USB PLL Clock 👻		
Clock divider:	2 Actual frequency [MHz]: 100		

- 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	Contraction of the local		
App Instance Name	App Pin Name	Pin Number (Port)	60
▲ SCU_CLK	external_clock_pin	#94(P1.15)	•
?		Solve And Save Res	et Close

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 observes 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)

```
#include <DAVE.h> //Declarations from DAVE Cod
e Generation (includes SFR declaration)
 volatile uint32_t TRAP_OSC_WDG_FLAG=0;
 volatile uint32_t TRAP_VC0_LOCK_FLAG=0;
 volatile uint32_t TRAP_USB_VC0_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_TRA
P TRAPSTAT SOSCWDGT Pos)
  {
      TRAP_OSC_WDG_FLAG++;
      XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_OSC_WD
G);
     // Add application code here for handling OSC
_HP oscillator watchdog trap
  }
  if((TRAP_FLAG & XMC_SCU_TRAP_VCO_LOCK) >> SCU_TR
AP_TRAPSTAT_SVCOLCKT_Pos)
  {
      TRAP_VC0_LOCK_FLAG++;
```

```
XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_VC0_L0
CK);
     // Add application code here for handling sys
tem VCO lock trap
  }
  if((TRAP_FLAG & XMC_SCU_TRAP_USB_VCO_LOCK) >> SC
U TRAP TRAPSTAT UVCOLCKT Pos)
  {
      TRAP_USB_VC0_FLAG++;
      XMC_SCU_TRAP_ClearStatus(XMC_SCU_TRAP_USB_VC
0_LOCK);
     // Add application code here for handling USB
VCO lock trap
  }
  if((TRAP FLAG & XMC SCU TRAP ULP WDG) >> SCU TRA
P TRAPSTAT ULPWDGT Pos)
  {
      TRAP ULP WDG FLAG++;
      XMC SCU TRAP ClearStatus(XMC SCU TRAP ULP WD
G);
     // Add application code here for handling OSC
_ULP oscillator watchdog trap
 }
}
11
// @brief main() - Application entry point
11
// <b>Details of function</b><br>
// This routine is the application entry point. I
t is invoked by the device startup code. It is res
ponsible for
// invoking the App initialization dispatcher rou
tine - DAVE_Init() and hosting the place-holder fo
r user application
// code.
11
 int main(void)
```

```
{
  DAVE_STATUS_t status;
  status = DAVE_Init(); // CLOCK_XMC4_Init() is c
alled from within DAVE_Init().
  if(status == DAVE_STATUS_FAILURE)
  {
   // Placeholder for error handler code. The whil
e loop below can be replaced with an user error ha
ndler
   XMC_DEBUG(("DAVE Apps initialization failed wit
h status %d\n", status));
  while(1U)
   {
  }
  }
  // Placeholder for user application code. The wh
ile loop below can be replaced with user applicati
on 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.

Home

Release History

Release History

Home				
Data Structures	Data Structure Index	Data Fields		
Data Structures				
Here are the data structures with brief descriptions:				
CLOCK_XMC4 Configuration structure for CLOCK_XMC4 APP				

Home			
Data Structures	Data Structure Index	Data Fields	
CLOCK_X Reference	MC4 Struct		Data Fields

Detailed Description

Configuration structure for CLOCK_XMC4 APP.

Definition at line **121** of file **CLOCK_XMC4.h**.

#include <CLOCK_XMC4.h>

Data Fields

bool init_status

Field Documentation

bool CLOCK_XMC4::init_status

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



Hon	ne			
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_XMC4

Hon	ne			
Data	Structures	Data Structure Index	Data Fields	
All	Variables			

• init_status : CLOCK_XMC4



File Lis	st
File List	Globals
Home	

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

```
CLOCK_XMC4.c [code]
CLOCK_XMC4.h [code]
```

Home

File List

Globals

Functions

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

DAVE_APP_VERSION_t	CLOCK_XMC4_GetAppVersion (void) Get CLOCK_XMC4 APP version.
CLOCK_XMC4_STATUS_t	CLOCK_XMC4_Init (CLOCK_XMC4_t *handle) Initializes a CLOCK_XMC4 APP instance.
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.
void	CLOCK_XMC4_StepSystemPIIFrequency (uint32_t kdiv) API for ramping down the system PLL clock frequency.

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:

```
#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:

```
#include <DAVE.h>
int main(void)
{
    DAVE_STATUS_t init_status;
    uint32_t kdiv = 10U; // (K2DIV+1) value for s
caling 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 co
nsumption by scaling down the system PLL clock fr
equency
    CLOCK_XMC4_StepSystemPllFrequency(kdiv); // f
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 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**.

Go to the source code of this file.

Home

File List

Globals

Data Structures

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.

```
#include <xmc_scu.h> #include <xmc_gpio.h>
#include <DAVE_Common.h>
#include "clock_xmc4_conf.h"
#include "clock_xmc4_extern.h"
```

Data Structures

struct CLOCK_XMC4

Configuration structure for **CLOCK_XMC4** APP. More...

Typedefs

typedef struct CLOCK_XMC4 CLOCK_XMC4_t Configuration structure for CLOCK_XMC4 APP.

Functions

DAVE_APP_VERSION_t	CLOCK_XMC4_GetAppVersion (void) Get CLOCK_XMC4 APP version.
CLOCK_XMC4_STATUS_t	CLOCK_XMC4_Init (CLOCK_XMC4_t *handle) Initializes a CLOCK_XMC4 APP instance.
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.
void	CLOCK_XMC4_StepSystemPliFrequency (uint32_t kdiv) API for ramping down the system PLL clock frequency.
enum	CLOCK_XMC4_STATUS { CLOCK_XMC4_STATUS_SUCCESS = 0U CLOCK_XMC4_STATUS_FAILURE = 1U }

Go to the source code of this file.

Hon	ne					
File L	ist	Globals				
All	Fu	nctions	Typedefs	Enumerations	Enumerator	

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

Hom	ne					
File L	.ist	Globals	_			
All	Fu	inctions	Typedefs	Enumerations	Enumerator	

- 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

Hon	ne					
File L	.ist	Globals				
All	Fu	nctions	Typedefs	Enumerations	Enumerator	

• CLOCK_XMC4_t : CLOCK_XMC4.h



Hon	ne					
File L	.ist	Globals				
All	Fu	nctions	Typedefs	Enumerations	Enumerator	

• CLOCK_XMC4_STATUS : CLOCK_XMC4.h



Hon	ne					
File L	.ist	Globals				
All	Fu	inctions	Typedefs	Enumerations	Enumerator	

- CLOCK_XMC4_STATUS_FAILURE : CLOCK_XMC4.h
- CLOCK_XMC4_STATUS_SUCCESS : CLOCK_XMC4.h

Home

File List

Globals

CLOCK_XMC4.h

Go to the documentation of this file.

```
00001
00059 #ifndef CLOCK XMC4 H
00060 #define CLOCK XMC4 H
00061
00063 * HEADER FILES
  00064
00065 #include <xmc scu.h>
00066 #include <xmc_gpio.h>
00067 #include <DAVE_Common.h>
00068 #include "clock_xmc4_conf.h"
00069
  00070
00071 * MACROS
  00072
*****************************
```

```
00073 #if (!((XMC_LIB_MAJOR_VERSION == 2U) && \
         (XMC_LIB_MINOR_VERSION >= 0U) && ∖
00074
       (XMC_LIB_PATCH_VERSION >= OU)))
00075
00076 #error "CLOCK_XMC4 requires XMC Peripheral L
ibrary v2.0.0 or higher"
00077 #endif
00078
    00091
00092 * ENUMS
00098 /*
00099 * @brief enumeration for CLOCK_XMC4 APP
00100 */
00101 typedef enum CLOCK XMC4 STATUS
00102 {
00103 CLOCK_XMC4_STATUS_SUCCESS = OU,
00104 CLOCK XMC4 STATUS FAILURE = 1U
00105 } CLOCK_XMC4_STATUS_t;
00106
00112 * DATA STRUCTURES
**************************/
00121 typedef struct CLOCK_XMC4
00122 {
00123 bool init_status;
00124 } CLOCK_XMC4_t;
00129 #ifdef __cplusplus
00130 extern "C" {
```

```
00131 #endif
00133 * API Prototypes
     00134
**********************/
00173 DAVE_APP_VERSION_t CLOCK_XMC4_GetAppVersion(
void);
00174
00209 CLOCK XMC4 STATUS t CLOCK XMC4 Init(CLOCK XM
C4 t *handle);
00210
00211 #ifdef CLOCK XMC4 OSCHP ENABLED
00212
00224 uint32_t OSCHP_GetFrequency(void);
00225 #endif
00226
00265 void CLOCK_XMC4_StepSystemPllFrequency(uint3
2 t kdiv);
00269 #ifdef __cplusplus
00270 }
00271 #endif
00272
00273 /* Include APP extern declaration file */
00274 #include"clock_xmc4_extern.h"
00275
00276 #endif /* End of CLOCK XMC4 H */
4
                                    +
```

Home

File List

Globals

CLOCK_XMC4.c

Go to the documentation of this file.

00001 00062 /************************************
00064 ************************************
00065 #include "clock_xmc4.h" 00066
00067 /************************************
* * * * * * * * * * * * * * * * * * * *
00068 * MACROS 00069 **********************************

00070 00071 /************************************
* * * * * * * * * * * * * * * * * * * *
00072 * LOCAL DATA 00073 **********************************

00074 * LOCAL ROUTINES 00076 00077 00078 00080 * API IMPLEMENTATION 00082 /* API to retrieve version of the APP */ 00083 DAVE APP VERSION t CLOCK XMC4 GetAppVersion(void) 00084 { 00085 DAVE_APP_VERSION_t version; version.major = (uint8_t)CLOCK_XMC4_MAJOR_ 00086 VERSION; version.minor = (uint8_t)CLOCK_XMC4_MINOR_ 00087 VERSION; 00088 version.patch = (uint8_t)CLOCK_XMC4_PATCH_ VERSION; 00089 00090 return (version); 00091 } 00092 00093 /* 00094 * API to initialize the CLOCK XMC4 APP TRAP events */ 00095 00096 CLOCK_XMC4_STATUS_t CLOCK_XMC4_Init(CLOCK_XM C4_t *handle)

```
00097 {
        CLOCK_XMC4_STATUS_t status = CLOCK_XMC4_ST
00098
ATUS_SUCCESS;
00099
00100
        XMC ASSERT("CLOCK XMC4 APP handle function
pointer uninitialized", (handle != NULL));
00101
00102
        handle->init_status = true;
00103
        return (status);
00104
00105 }
00106
00107 #ifdef CLOCK XMC4 OSCHP ENABLED
00108 /* API to retrieve high precision external
oscillator frequency */
00109 uint32_t OSCHP_GetFrequency(void)
00110 {
      return (CLOCK_XMC4_OSCHP_FREQUENCY);
00111
00112 }
00113 #endif
00114
00115 /* API for ramping down the system PLL cloc
k frequency */
00116 void CLOCK_XMC4_StepSystemPllFrequency(uint3
2_t kdiv)
00117 {
      XMC_ASSERT("Incorrect kdiv value", ((kdiv
00118
>= 1) && (kdiv >= 128)));
       XMC_SCU_CLOCK_StepSystemPllFrequency(kdiv)
00119
00120 }
```