

```

#include <stdio.h>
#include <stdlib.h>
#include "uart.h"
#include <MKL25Z4.H>

// MYCODE

int UART_configure(void)
{
    SIM_SCGC4 |= 0x400; /* enable clock for UART0 */
    SIM_SOPT2 |= 0x4000000; /* use FLL output for UART Baud rate generator*/
    UART0->C2 = 0;
    SIM_SCGC5 |= 0x0200; /* enable clock for PORTA */
    PORTA->PCR[2] = 0x0200;
}

int UART_send(char *data_send)
{
    while (1)
    {
        while(!(UART0->S1 & 0x80))
        {
            } /* wait for transmit buffer empty */

        UART0->D = *data_send;
    }
}

int UART_send_n(char *data_send_n, int length)
{
    while (1)

```

```

{

{for (int i=0;i<=length-1;i++)

    while(!(UART0->S1 & 0x80))

    {

} /* wait for transmit buffer empty */

UART0->D = *data_send_n;

data_send_n++;

}

}

int UART_receive(char *data_receive)

{

    char c;

    while (1)

    {

        while(!(UART0->S1 & 0x20))

        {

} /* wait for receive buffer full */

*data_receive = UART2->D ; /* read the char received */

}

}

int UART_receive_n(char *data_receive_n,int length)

{

    char c;

    while (1)

    {

        for(int i=0;i<=length-1;i++)

```

```
{while(!(UART0->S1 & 0x20))
{
} /* wait for receive buffer full */

*data_receive_n = UART2->D ; /* read the char received */

data_receive_n++;

}

}

/*
** #####
** Processors:      MKL25Z128FM4
**
**                  MKL25Z128FT4
**
**                  MKL25Z128LH4
**
**                  MKL25Z128VLK4
**
**
** Compilers:      Keil ARM C/C++ Compiler
**
**                  Freescale C/C++ for Embedded ARM
**
**                  GNU C Compiler
**
**                  GNU C Compiler - CodeSourcery Sourcery G++
**
**                  IAR ANSI C/C++ Compiler for ARM
**
**
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012
**
** Version:        rev. 2.5, 2015-02-19
**
** Build:          b150220
**
**
** Abstract:
**
** Provides a system configuration function and a global variable that
** contains the system frequency. It configures the device and initializes
** the oscillator (PLL) that is part of the microcontroller device.
```

**

** Copyright (c) 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com
** mail: support@freescale.com
**
** Revisions:
** - rev. 1.0 (2012-06-13)
** Initial version.
** - rev. 1.1 (2012-06-21)
** Update according to reference manual rev. 1.
** - rev. 1.2 (2012-08-01)
** Device type UARTLP changed to UART0.
** - rev. 1.3 (2012-10-04)
** Update according to reference manual rev. 3.
** - rev. 1.4 (2012-11-22)
** MCG module - bit LOLS in MCG_S register renamed to LOLS0.
** NV registers - bit EZPORT_DIS in NV_FOPT register removed.
** - rev. 1.5 (2013-04-05)
** Changed start of doxygen comment.
** - rev. 2.0 (2013-10-29)
** Register accessor macros added to the memory map.
** Symbols for Processor Expert memory map compatibility added to the memory map.
** Startup file for gcc has been updated according to CMSIS 3.2.
** System initialization updated.
** - rev. 2.1 (2014-07-16)
** Module access macro module_BASES replaced by module_BASE_PTRS.
** System initialization and startup updated.
** - rev. 2.2 (2014-08-22)
** System initialization updated - default clock config changed.
** - rev. 2.3 (2014-08-28)
** Update of startup files - possibility to override DefaultISR added.

```
** - rev. 2.4 (2014-10-14)
**     Interrupt INT_LPTimer renamed to INT_LPTMR0.
** - rev. 2.5 (2015-02-19)
**     Renamed interrupt vector LLW to LLWU.
**
** #####
*/

```

```
/*!
* @file MKL25Z4
* @version 2.5
* @date 2015-02-19
* @brief Device specific configuration file for MKL25Z4 (implementation file)
*
* Provides a system configuration function and a global variable that contains
* the system frequency. It configures the device and initializes the oscillator
* (PLL) that is part of the microcontroller device.
*/

```

```
#include <stdint.h>
#include "MKL25Z4.h"

/*
----- --
-- Core clock
----- */

```

```
uint32_t SystemCoreClock = DEFAULT_SYSTEM_CLOCK;
```

```

/*
-----  

-- SystemInit()  

----- */
  
  

void SystemInit (void) {  

    #if (DISABLE_WDOG)  

        /* SIM_COPC: COPT=0,COPCLKS=0,COPW=0 */  

        SIM->COPC = (uint32_t)0x00u;  

    #endif /* (DISABLE_WDOG) */  

  

    #ifdef CLOCK_SETUP  

        if((RCM->SRS0 & RCM_SRS0_WAKEUP_MASK) != 0x00U)  

        {  

            if((PMC->REGSC & PMC_REGSC_ACKISO_MASK) != 0x00U)  

            {  

                PMC->REGSC |= PMC_REGSC_ACKISO_MASK; /* Release hold with ACKISO: Only has an effect if  

recovering from VLLSx.*/  

            }  

        }  

    }  

  

/* Power mode protection initialization */  

  

#ifdef SYSTEM_SMC_PMPROT_VALUE  

    SMC->PMPROT = SYSTEM_SMC_PMPROT_VALUE;  

#endif  

  

/* System clock initialization */  

/* Internal reference clock trim initialization */  

  

#if defined(SLOW_TRIM_ADDRESS)

```

```

if ( *((uint8_t*)SLOW_TRIM_ADDRESS) != 0xFFU ) { /* Skip if non-volatile flash memory
is erased */

    MCG->C3 = *((uint8_t*)SLOW_TRIM_ADDRESS);

#endif /* defined(SLOW_TRIM_ADDRESS) */

#if defined(SLOW_FINE_TRIM_ADDRESS)

    MCG->C4 = (MCG->C4 & ~(MCG_C4_SCFTRIM_MASK)) | ((*((uint8_t*) SLOW_FINE_TRIM_ADDRESS)) &
& MCG_C4_SCFTRIM_MASK);

#endif

#if defined(FAST_TRIM_ADDRESS)

    MCG->C4 = (MCG->C4 & ~(MCG_C4_FCTRIM_MASK)) |((*((uint8_t*) FAST_TRIM_ADDRESS)) &
MCG_C4_FCTRIM_MASK);

#endif

#if defined(SLOW_TRIM_ADDRESS)

}

#endif /* defined(SLOW_TRIM_ADDRESS) */

/* Set system prescalers and clock sources */

SIM->CLKDIV1 = SYSTEM_SIM_CLKDIV1_VALUE; /* Set system prescalers */

SIM->SOPT1 = ((SIM->SOPT1) & (uint32_t)(~(SIM_SOPT1_OSC32KSEL_MASK))) |  

((SYSTEM_SIM_SOPT1_VALUE) & (SIM_SOPT1_OSC32KSEL_MASK)); /* Set 32 kHz clock source  

(ERCLK32K) */

SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_PLLFLLSEL_MASK))) |  

((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_PLLFLLSEL_MASK)); /* Selects the high frequency clock  

for various peripheral clocking options. */

SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_TPMSRC_MASK))) |  

((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_TPMSRC_MASK)); /* Selects the clock source for the TPM  

counter clock. */

#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FBI) || (MCG_MODE ==  

MCG_MODE_BLPI))

/* Set MCG and OSC */

#if (((SYSTEM_OSCO_CR_VALUE) & OSC_CR_ERCLKEN_MASK) != 0x00U) ||
((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKENO_MASK) != 0x00U))

/* SIM_SCGC5: PORTA=1 */

```

```

SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

/* PORTA_PCR18: ISF=0,MUX=0 */

PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFS0_MASK) != 0x00U) {

/* PORTA_PCR19: ISF=0,MUX=0 */

PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

}

#endif

MCG->SC = SYSTEM_MCG_SC_VALUE; /* Set SC (fast clock internal reference divider) */

MCG->C1 = SYSTEM_MCG_C1_VALUE; /* Set C1 (clock source selection, FLL ext. reference divider,
int. reference enable etc.) */

/* Check that the source of the FLL reference clock is the requested one. */

if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {

while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {

}

} else {

while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {

}

}

MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set C2 (freq. range, ext.
and int. reference selection etc.; low power bit is set later) */

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK | MCG_C4_SCFTRIM_MASK)); /*

Set C4 (FLL output; trim values not changed) */

OSCO->CR = SYSTEM_OSCO_CR_VALUE; /* Set OSC_CR (OSCERCLK enable, oscillator capacitor load)
*/
#endif

#if (MCG_MODE == MCG_MODE_BLPI)

/* BLPI specific */

MCG->C2 |= (MCG_C2_LP_MASK); /* Disable FLL and PLL in bypass mode */

#endif

```

```

#else /* MCG_MODE */

/* Set MCG and OSC */

/* SIM_SCGC5: PORTA=1 */

SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

/* PORTA_PCR18: ISF=0,MUX=0 */

PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFS0_MASK) != 0x00U) {

/* PORTA_PCR19: ISF=0,MUX=0 */

PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

}

MCG->SC = SYSTEM_MCG_SC_VALUE; /* Set SC (fast clock internal reference divider) */

MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set C2 (freq. range, ext.
and int. reference selection etc.; low power bit is set later) */

OSCO->CR = SYSTEM_OSCO_CR_VALUE; /* Set OSC_CR (OSCERCLK enable, oscillator capacitor load)
*/
#endif /* MCG_MODE == MCG_MODE_PEE */

MCG->C1 = (SYSTEM_MCG_C1_VALUE) | MCG_C1_CLKS(0x02); /* Set C1 (clock source selection, FLL
ext. reference divider, int. reference enable etc.) - PBE mode */

#ifndef MCG_MODE_PEE
MCG->C1 = SYSTEM_MCG_C1_VALUE; /* Set C1 (clock source selection, FLL ext. reference divider,
int. reference enable etc.) */
#endif /* MCG_MODE_PEE */

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFS0_MASK) != 0x00U) {

while((MCG->S & MCG_S_OSCINIT0_MASK) == 0x00U) { /* Check that the oscillator is running */

}

/* Check that the source of the FLL reference clock is the requested one. */

if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {

while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {

```

```

    }

} else {

    while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {

    }

}

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK | MCG_C4_SCFTRIM_MASK)); /*  

Set C4 (FLL output; trim values not changed) */

#endif /* MCG_MODE */

/* Common for all MCG modes */

/* PLL clock can be used to generate clock for some devices regardless of clock generator  

(MCGOUTCLK) mode. */

MCG->C5 = (SYSTEM_MCG_C5_VALUE) & (uint8_t)(~(MCG_C5_PLLCLKENO_MASK)); /* Set C5 (PLL  

settings, PLL reference divider etc.) */

MCG->C6 = (SYSTEM_MCG_C6_VALUE) & (uint8_t)~(MCG_C6_PLLS_MASK); /* Set C6 (PLL select, VCO  

divider etc.) */

if ((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKENO_MASK) {

    MCG->C5 |= MCG_C5_PLLCLKENO_MASK; /* PLL clock enable in mode other than PEE or PBE */

}

/* BLPE, PEE and PBE MCG mode specific */

#if (MCG_MODE == MCG_MODE_BLPE)

    MCG->C2 |= (MCG_C2_LP_MASK); /* Disable FLL and PLL in bypass mode */

#elif ((MCG_MODE == MCG_MODE_PBE) || (MCG_MODE == MCG_MODE_PEE))

    MCG->C6 |= (MCG_C6_PLLS_MASK); /* Set C6 (PLL select, VCO divider etc.) */

    while((MCG->S & MCG_S_LOCK0_MASK) == 0x00U) { /* Wait until PLL is locked*/

}

#endif (MCG_MODE == MCG_MODE_PEE)

```

```

MCG->C1 &= (uint8_t)~(MCG_C1_CLKS_MASK);

#endif

#endif

#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FEE))

while((MCG->S & MCG_S_CLKST_MASK) != 0x00U) { /* Wait until output of the FLL is selected */

}

/* Use LPTMR to wait for 1ms for FLL clock stabilization */

SIM_SCGC5 |= SIM_SCGC5_LPTMR_MASK; /* Allow software control of LPMTR */

LPTMR0->CMR = LPTMR_CMR_COMPARE(0); /* Default 1 LPO tick */

LPTMR0->CSR = (LPTMR_CSR_TCF_MASK | LPTMR_CSR_TPS(0x00));

LPTMR0->PSR = (LPTMR_PSR_PCS(0x01) | LPTMR_PSR_PBYP_MASK); /* Clock source: LPO, Prescaler bypass enable */

LPTMR0->CSR = LPTMR_CSR_TEN_MASK; /* LPMTR enable */

while((LPTMR0_CSR & LPTMR_CSR_TCF_MASK) == 0u) {

}

LPTMR0_CSR = 0x00; /* Disable LPTMR */

SIM_SCGC5 &= (uint32_t)~(uint32_t)SIM_SCGC5_LPTMR_MASK;

#elif ((MCG_MODE == MCG_MODE_FBI) || (MCG_MODE == MCG_MODE_BLPI))

while((MCG->S & MCG_S_CLKST_MASK) != 0x04U) { /* Wait until internal reference clock is selected as MCG output */

}

#elif ((MCG_MODE == MCG_MODE_FBE) || (MCG_MODE == MCG_MODE_PBE) || (MCG_MODE == MCG_MODE_BLPE))

while((MCG->S & MCG_S_CLKST_MASK) != 0x08U) { /* Wait until external reference clock is selected as MCG output */

}

#elif (MCG_MODE == MCG_MODE_PEE)

while((MCG->S & MCG_S_CLKST_MASK) != 0x0CU) { /* Wait until output of the PLL is selected */

}

#endif

```

```

#if (((SYSTEM_SMC_PMCTRL_VALUE) & SMC_PMCTRL_RUNM_MASK) == (0x02U <<
SMC_PMCTRL_RUNM_SHIFT))

    SMC->PMCTRL = (uint8_t)((SYSTEM_SMC_PMCTRL_VALUE) & (SMC_PMCTRL_RUNM_MASK)); /*

Enable VLPR mode */

    while(SMC->PMSTAT != 0x04U) {      /* Wait until the system is in VLPR mode */

}

#endif

/* PLL loss of lock interrupt request initialization */

if (((SYSTEM_MCG_C6_VALUE) & MCG_C6_LOLIE0_MASK) != 0U) {

    NVIC_EnableIRQ(MCG_IRQn);      /* Enable PLL loss of lock interrupt request */

}

#endif

}

/* -----
-- SystemCoreClockUpdate()
----- */

void SystemCoreClockUpdate (void) {

    uint32_t MCGOUTClock;          /* Variable to store output clock frequency of the MCG module */

    uint16_t Divider;

    if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x00U) {

        /* Output of FLL or PLL is selected */

        if ((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U) {

            /* FLL is selected */

            if ((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U) {

                /* External reference clock is selected */

```

```

MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG clock */

if ((MCG->C2 & MCG_C2_RANGE0_MASK) != 0x00U) {

    switch (MCG->C1 & MCG_C1_FRDIV_MASK) {

        case 0x38U:

            Divider = 1536U;

            break;

        case 0x30U:

            Divider = 1280U;

            break;

        default:

            Divider = (uint16_t)(32LU << ((MCG->C1 & MCG_C1_FRDIV_MASK) >> MCG_C1_FRDIV_SHIFT));

            break;

    }

} else /* ((MCG->C2 & MCG_C2_RANGE_MASK) != 0x00U) */

    Divider = (uint16_t)(1LU << ((MCG->C1 & MCG_C1_FRDIV_MASK) >> MCG_C1_FRDIV_SHIFT));

}

MCGOUTClock = (MCGOUTClock / Divider); /* Calculate the divided FLL reference clock */

} else { /* (!((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U)) */

    MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* The slow internal reference clock is selected */

} /* (!((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U)) */

/* Select correct multiplier to calculate the MCG output clock */

switch (MCG->C4 & (MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) {

    case 0x00U:

        MCGOUTClock *= 640U;

        break;

    case 0x20U:

        MCGOUTClock *= 1280U;

        break;

    case 0x40U:

```

```

MCGOUTClock *= 1920U;
break;

case 0x60U:
MCGOUTClock *= 2560U;
break;

case 0x80U:
MCGOUTClock *= 732U;
break;

case 0xA0U:
MCGOUTClock *= 1464U;
break;

case 0xC0U:
MCGOUTClock *= 2197U;
break;

case 0xE0U:
MCGOUTClock *= 2929U;
break;

default:
break;
}

} else { /* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */

/* PLL is selected */

Divider = (((uint16_t)MCG->C5 & MCG_C5_PRDIV0_MASK) + 0x01U);
MCGOUTClock = (uint32_t)(CPU_XTAL_CLK_HZ / Divider); /* Calculate the PLL reference clock */
Divider = (((uint16_t)MCG->C6 & MCG_C6_VDIV0_MASK) + 24U);
MCGOUTClock *= Divider; /* Calculate the MCG output clock */

} /* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */

} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x40U) {

/* Internal reference clock is selected */

```

```

if ((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U) {
    MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* Slow internal reference clock selected */
} else { /* (!((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U)) */
    Divider = (uint16_t)(0x01LU << ((MCG->SC & MCG_SC_FCRDIV_MASK) >> MCG_SC_FCRDIV_SHIFT));
    MCGOUTClock = (uint32_t) (CPU_INT_FAST_CLK_HZ / Divider); /* Fast internal reference clock
selected */
} /* (!((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U)) */

} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U) {
    /* External reference clock is selected */
    MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG clock */
} else { /* (!((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U)) */
    /* Reserved value */
    return;
} /* (!((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U)) */

SystemCoreClock = (MCGOUTClock / (0x01U + ((SIM->CLKDIV1 & SIM_CLKDIV1_OUTDIV1_MASK) >>
SIM_CLKDIV1_OUTDIV1_SHIFT)));
}

/*********************//**/

* @file core_cmSimd.h
* @brief CMSIS Cortex-M SIMD Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*
*****/
/* Copyright (c) 2009 - 2014 ARM LIMITED

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )  
#pragma system_include /* treat file as system include file for MISRA check */  
#endif
```

```

#ifndef __CORE_CMSIMD_H
#define __CORE_CMSIMD_H


#ifndef __cplusplus
extern "C" {

#endif


//*****************************************************************************
*          Hardware Abstraction Layer
*****/


/* ##### Compiler specific Intrinsics #####
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
Access to dedicated SIMD instructions

@{
*/
#ifndef __CC_ARM /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#define __SADD8      __sadd8
#define __QADD8      __qadd8
#define __SHADD8     __shadd8
#define __UADD8      __uadd8
#define __UQADD8     __uqadd8
#define __UHADD8     __uhadd8
#define __SSUB8      __ssub8
#define __QSUB8      __qsub8

```

```
#define __SHSUB8          __shsub8
#define __USUB8           __usub8
#define __UQSUB8          __uqsub8
#define __UHSUB8          __uhsub8
#define __SADD16           __sadd16
#define __QADD16           __qadd16
#define __SHADD16          __shadd16
#define __UADD16           __uadd16
#define __UQADD16          __uqadd16
#define __UHADD16          __uhadd16
#define __SSUB16            __ssub16
#define __QSUB16            __qsub16
#define __SHSUB16          __shsub16
#define __USUB16           __usub16
#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX              __sasx
#define __QASX              __qasx
#define __SHASX             __shasx
#define __UASX              __uasx
#define __UQASX             __uqasx
#define __UHASX             __uhasx
#define __SSAX               __ssax
#define __QSAX               __qsax
#define __SHSAX              __shsax
#define __USAX               __usax
#define __UQSAX              __uqsax
#define __UHSAX              __uhsax
#define __USAD8              __usad8
```

```

#define __USADA8           __usada8
#define __SSAT16            __ssat16
#define __USAT16            __usat16
#define __UXTB16            __uxtb16
#define __UXTAB16           __uxtab16
#define __SXTB16            __sxtb16
#define __SXTAB16           __sxtab16
#define __SMUAD             __smuad
#define __SMUADX            __smuadx
#define __SMLAD              __smlad
#define __SMLADX             __smladx
#define __SMLALD             __smlald
#define __SMLALDX            __smlaldx
#define __SMUSD               __smusd
#define __SMUSDX             __smusdx
#define __SMLSD               __smlsd
#define __SMLSDX              __smlsdx
#define __SMLS LD             __smlsld
#define __SMLS LDX            __smlsldx
#define __SEL                 __sel
#define __QADD                __qadd
#define __QSUB                __qsub

#define __PKHBT(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      & 0x0000FFFFUL) | \
                                (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      & 0xFFFF0000UL) | \
                                (((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL) )

```

```

#define __SMMLA(ARG1,ARG2,ARG3)      ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                         ((int64_t)(ARG3) << 32)    ) >> 32))

#elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD16(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
}
```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSAX(uint32_t op1, uint32_t op2)
{
```

```

    uint32_t result;

    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USADA8(uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#define __SSAT16(ARG1,ARG2) \
{ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
}

```

```

#define __USAT16(ARG1,ARG2) \
{ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));

```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUAD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUADX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLAD (uint32_t op1, uint32_t op2,
    uint32_t op3)
```

```

{

    uint32_t result;

    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLADX (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLALD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian

```

```

__ASM volatile ("smalld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUSD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

```

```
__ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUSDX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
    return(result);
}
```

```

}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLSLD (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLSLDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

```

```

#ifndef __ARMEB__ // Little endian

__ASM volatile ("smisIdx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else // Big endian

__ASM volatile ("smisIdx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);

}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SEL (uint32_t op1, uint32_t op2)
{
    uint32_t result;

__ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __QADD(uint32_t op1, uint32_t op2)
{
    uint32_t result;

__ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __QSUB(uint32_t op1, uint32_t op2)

```

```

{

    uint32_t result;

    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

#define __PKHBT(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3)); \
    __RES; \
})

#define __PKHTB(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2)); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3)); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMMLA (int32_t op1, int32_t op2, int32_t
op3)
{
    int32_t result;
}

```

```
__ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" (op2), "r" (op3) );
return(result);
}

#ifndef CMSIS_DSP_H
#define CMSIS_DSP_H

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/* not yet supported */

#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD_intrinsics */
```

```
#ifdef __cplusplus
}

#endif /* __CORE_CMSIMD_H */
/**********************************************************//*
 * @file core_cm4.h
 * @brief CMSIS Cortex-M4 Core Peripheral Access Layer Header File
 * @version V4.10
 * @date 18. March 2015
 *
 * @note
 *
 *****/
/* Copyright (c) 2009 - 2015 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )  
#pragma system_include /* treat file as system include file for MISRA check */  
#endif
```

```
#ifndef __CORE_CM4_H_GENERIC  
#define __CORE_CM4_H_GENERIC
```

```
#ifdef __cplusplus  
extern "C" {  
#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions  
CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.

*/

```
/****************************************************************************
 * CMSIS definitions
 ****/
/** \ingroup Cortex_M4
 @{
 */

/* CMSIS CM4 definitions */

#define __CM4_CMSIS_VERSION_MAIN (0x04)           /*!< [31:16] CMSIS HAL main version
*/
#define __CM4_CMSIS_VERSION_SUB  (0x00)             /*!< [15:0] CMSIS HAL sub version */
#define __CM4_CMSIS_VERSION     ((__CM4_CMSIS_VERSION_MAIN << 16) | \
                                __CM4_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */

#define __CORTEX_M      (0x04)           /*!< Cortex-M Core */
                                         /* */

#if defined (__CC_ARM)
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */

```

```

#define __STATIC_INLINE static __inline

#elif defined ( __GNUC__ )

#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */

#define __STATIC_INLINE static inline


#elif defined ( __ICCARM__ )

#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */

#define __STATIC_INLINE static inline


#elif defined ( __TMS470__ )

#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline


#elif defined ( __TASKING__ )

#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline


#elif defined ( __CSMC__ )

#define __packed

#define __ASM      __asm           /*!< asm keyword for COSMIC Compiler */
#define __INLINE   inline          /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */

#define __STATIC_INLINE static inline

```

```

#endif

/** __FPU_USED indicates whether an FPU is used or not.

For this, __FPU_PRESENT has to be checked prior to making use of FPU specific registers and
functions.

*/
#ifndef __FPU_USED
#define __FPU_USED 0

#if defined (__CC_ARM)
    #if defined __TARGET_FPU_VFP
        #if (__FPU_PRESENT == 1)
            #define __FPU_USED 1
        #else
            #warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
            #define __FPU_USED 0
        #endif
    #else
        #define __FPU_USED 0
    #endif
#endif

#elif defined (__GNUC__)
    #if defined (__VFP_FP__) && !defined(__SOFTFP__)
        #if (__FPU_PRESENT == 1)
            #define __FPU_USED 1
        #else
            #warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
            #define __FPU_USED 0
        #endif
    #else
        #define __FPU_USED 0
    #endif

```

```
#endif

#if defined ( __ICCARM__ )

#if defined __ARMVFP__
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED    0
#endif
#else
#define __FPU_USED    0
#endif

#endif

#if defined ( __TMS470__ )

#if defined __TI_VFP_SUPPORT__
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED    0
#endif
#else
#define __FPU_USED    0
#endif

#endif

#if defined ( __TASKING__ )
```

```

#if defined __FPU_VFP__
    #if (__FPU_PRESENT == 1)
        #define __FPU_USED    1
    #else
        #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
        #define __FPU_USED    0
    #endif
#else
    #define __FPU_USED    0
#endif

#elif defined ( __CSMC__ )           /* Cosmic */
    #if ( __CSMC__ & 0x400)          // FPU present for parser
        #if (__FPU_PRESENT == 1)
            #define __FPU_USED    1
        #else
            #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
            #define __FPU_USED    0
        #endif
    #else
        #define __FPU_USED    0
    #endif
#endif

#include <stdint.h>             /* standard types definitions      */
#include <core_cmInstr.h>          /* Core Instruction Access        */
#include <core_cmFunc.h>           /* Core Function Access          */
#include <core_cmSimd.h>           /* Compiler specific SIMD Intrinsics */

```

```
#ifdef __cplusplus
}

#endif /* __CORE_CM4_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM4_H_DEPENDANT
#define __CORE_CM4_H_DEPENDANT

#ifndef __cplusplus
extern "C" {

#endif

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
#ifndef __CM4_REV
#define __CM4_REV      0x0000
#warning "__CM4_REV not defined in device header file; using default!"
#endif

#ifndef __FPU_PRESENT
#define __FPU_PRESENT    0
#warning "__FPU_PRESENT not defined in device header file; using default!"
#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT    0

```

```

#warning "__MPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      4
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif

#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
\defgroup CMSIS_glob_defs CMSIS Global Defines

<strong>IO Type Qualifiers</strong> are used
\li to specify the access to peripheral variables.
\li for automatic generation of peripheral register debug information.
*/
#ifndef __cplusplus
#define __I volatile      /*!< Defines 'read only' permissions      */
#else
#define __I volatile const /*!< Defines 'read only' permissions      */
#endif
#define __O volatile      /*!< Defines 'write only' permissions     */
#define __IO volatile     /*!< Defines 'read / write' permissions   */

```

```
/*@} end of group Cortex_M4 */
```

```
*****
```

```
* Register Abstraction
```

Core Register contain:

- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core Debug Register
- Core MPU Register
- Core FPU Register

```
*****/
```

```
/** \defgroup CMSIS_core_register Defines and Type Definitions
```

\\brief Type definitions and defines for Cortex-M processor based devices.

```
*/
```

```
/** \ingroup CMSIS_core_register
```

```
\\defgroup CMSIS_CORE Status and Control Registers
```

\\brief Core Register type definitions.

```
@{
```

```
*/
```

```
/** \\brief Union type to access the Application Program Status Register (APSR).
```

```
*/
```

```
typedef union
```

```

{

struct
{

    uint32_t _reserved0:16;      /*!< bit: 0..15 Reserved          */
    uint32_t GE:4;              /*!< bit: 16..19 Greater than or Equal flags   */
    uint32_t _reserved1:7;      /*!< bit: 20..26 Reserved          */
    uint32_t Q:1;               /*!< bit: 27 Saturation condition flag        */
    uint32_t V:1;               /*!< bit: 28 Overflow condition code flag     */
    uint32_t C:1;               /*!< bit: 29 Carry condition code flag       */
    uint32_t Z:1;               /*!< bit: 30 Zero condition code flag        */
    uint32_t N:1;               /*!< bit: 31 Negative condition code flag    */

} b;                      /*!< Structure used for bit access           */

uint32_t w;                /*!< Type used for word access             */

} APSR_Type;

```

```

/* APSR Register Definitions */

#define APSR_N_Pos            31          /*!< APSR: N Position */
#define APSR_N_Msk             (1UL << APSR_N_Pos)  /*!< APSR: N Mask */

#define APSR_Z_Pos            30          /*!< APSR: Z Position */
#define APSR_Z_Msk             (1UL << APSR_Z_Pos)  /*!< APSR: Z Mask */

#define APSR_C_Pos            29          /*!< APSR: C Position */
#define APSR_C_Msk             (1UL << APSR_C_Pos)  /*!< APSR: C Mask */

#define APSR_V_Pos            28          /*!< APSR: V Position */
#define APSR_V_Msk             (1UL << APSR_V_Pos)  /*!< APSR: V Mask */

#define APSR_Q_Pos            27          /*!< APSR: Q Position */

```

```

#define APSR_Q_Msk          (1UL << APSR_Q_Pos)           /*!< APSR: Q Mask */
#define APSR_GE_Pos          16                           /*!< APSR: GE Position */
#define APSR_GE_Msk          (0xFUL << APSR_GE_Pos)        /*!< APSR: GE Mask */

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

*/

typedef union

{

struct

{

 uint32_t ISR:9; /*!< bit: 0.. 8 Exception number */

 uint32_t _reserved0:23; /*!< bit: 9..31 Reserved */

 } b; /*!< Structure used for bit access */

 uint32_t w; /*!< Type used for word access */

} IPSR_Type;

/* IPSR Register Definitions */

```
#define IPSR_ISR_Pos          0           /*!< IPSR: ISR Position */
```

```
#define IPSR_ISR_Msk          (0x1FFUL /*<< IPSR_ISR_Pos*/)      /*!< IPSR: ISR Mask */
```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).

*/

typedef union

{

struct

{

```

    uint32_t ISR:9;           /*!< bit: 0..8 Exception number      */
    uint32_t _reserved0:7;     /*!< bit: 9..15 Reserved            */
    uint32_t GE:4;            /*!< bit: 16..19 Greater than or Equal flags */
    uint32_t _reserved1:4;     /*!< bit: 20..23 Reserved            */
    uint32_t T:1;              /*!< bit: 24 Thumb bit   (read 0)    */
    uint32_t IT:2;             /*!< bit: 25..26 saved IT state (read 0) */
    uint32_t Q:1;              /*!< bit: 27 Saturation condition flag */
    uint32_t V:1;              /*!< bit: 28 Overflow condition code flag */
    uint32_t C:1;              /*!< bit: 29 Carry condition code flag */
    uint32_t Z:1;              /*!< bit: 30 Zero condition code flag  */
    uint32_t N:1;              /*!< bit: 31 Negative condition code flag */

} b;                      /*!< Structure used for bit access      */

uint32_t w;                /*!< Type   used for word access       */

} xPSR_Type;

```

```

/* xPSR Register Definitions */

#define xPSR_N_Pos           31          /*!< xPSR: N Position */
#define xPSR_N_Msk          (1UL << xPSR_N_Pos) /*!< xPSR: N Mask */

#define xPSR_Z_Pos           30          /*!< xPSR: Z Position */
#define xPSR_Z_Msk          (1UL << xPSR_Z_Pos) /*!< xPSR: Z Mask */

#define xPSR_C_Pos           29          /*!< xPSR: C Position */
#define xPSR_C_Msk          (1UL << xPSR_C_Pos) /*!< xPSR: C Mask */

#define xPSR_V_Pos           28          /*!< xPSR: V Position */
#define xPSR_V_Msk          (1UL << xPSR_V_Pos) /*!< xPSR: V Mask */

#define xPSR_Q_Pos           27          /*!< xPSR: Q Position */

```

```

#define xPSR_Q_Msk          (1UL << xPSR_Q_Pos)           /*!< xPSR: Q Mask */

#define xPSR_IT_Pos          25                           /*!< xPSR: IT Position */
#define xPSR_IT_Msk          (3UL << xPSR_IT_Pos)         /*!< xPSR: IT Mask */

#define xPSR_T_Pos            24                           /*!< xPSR: T Position */
#define xPSR_T_Msk            (1UL << xPSR_T_Pos)          /*!< xPSR: T Mask */

#define xPSR_GE_Pos           16                           /*!< xPSR: GE Position */
#define xPSR_GE_Msk           (0xFUL << xPSR_GE_Pos)        /*!< xPSR: GE Mask */

#define xPSR_ISR_Pos          0                            /*!< xPSR: ISR Position */
#define xPSR_ISR_Msk          (0x1FFUL /*<< xPSR_ISR_Pos*/) /*!< xPSR: ISR Mask */

```

```

/** \brief Union type to access the Control Registers (CONTROL).

*/
typedef union
{
    struct
    {
        uint32_t nPRIV:1;           /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                         /*!< Structure used for bit access */
    uint32_t w;                  /*!< Type used for word access */
} CONTROL_Type;

```

```

/* CONTROL Register Definitions */

#define CONTROL_FPCA_Pos           2             /*!< CONTROL: FPCA Position */
#define CONTROL_FPCA_Msk          (1UL << CONTROL_FPCA_Pos)      /*!< CONTROL: FPCA
Mask */

#define CONTROL_SPSEL_Pos          1             /*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk         (1UL << CONTROL_SPSEL_Pos)      /*!< CONTROL: SPSEL
Mask */

#define CONTROL_nPRIV_Pos          0             /*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk        (1UL /*<< CONTROL_nPRIV_Pos*/)      /*!< CONTROL:
nPRIV Mask */

/*}@} end of group CMSIS_CORE */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
@{
*/

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[8];           /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */
    uint32_t RESERVED0[24];
    __IO uint32_t ICER[8];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
    uint32_t RSERVED1[24];
}

```

```

__IO uint32_t ISPR[8];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register      */
uint32_t RESERVED2[24];

__IO uint32_t ICPR[8];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register     */
uint32_t RESERVED3[24];

__IO uint32_t IABR[8];           /*!< Offset: 0x200 (R/W) Interrupt Active bit Register       */
uint32_t RESERVED4[56];

__IO uint8_t IP[240];            /*!< Offset: 0x300 (R/W) Interrupt Priority Register (8Bit wide) */
uint32_t RESERVED5[644];

__O uint32_t STIR;              /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register  */

} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */

#define NVIC_STIR_INTID_Pos          0                  /*!< STIR: INTLINESNUM Position */
#define NVIC_STIR_INTID_Msk          (0x1FUL/*<< NVIC_STIR_INTID_Pos*/)    /*!< STIR:
INTLINESNUM Mask */

/*@} end of group CMSIS_NVIC */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers
@{
*/

```

```

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct
{

```

```

__I uint32_t CPUID;          /*!< Offset: 0x000 (R/ ) CPUID Base Register           */
__IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
*/
__IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
__IO uint32_t AIRCR;          /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register   */
__IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register           */
__IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register     */
__IO uint8_t SHP[12];         /*!< Offset: 0x018 (R/W) System Handlers Priority Registers (4-7, 8-11,
12-15) */
__IO uint32_t SHCSR;          /*!< Offset: 0x024 (R/W) System Handler Control and State Register */
*/
__IO uint32_t CFSR;          /*!< Offset: 0x028 (R/W) Configurable Fault Status Register */
*/
__IO uint32_t HFSR;          /*!< Offset: 0x02C (R/W) HardFault Status Register           */
__IO uint32_t DFSR;          /*!< Offset: 0x030 (R/W) Debug Fault Status Register         */
__IO uint32_t MMFAR;          /*!< Offset: 0x034 (R/W) MemManage Fault Address Register    */
*/
__IO uint32_t BFAR;           /*!< Offset: 0x038 (R/W) BusFault Address Register           */
__IO uint32_t AFSR;           /*!< Offset: 0x03C (R/W) Auxiliary Fault Status Register     */
__I uint32_t PFR[2];          /*!< Offset: 0x040 (R/ ) Processor Feature Register         */
__I uint32_t DFR;           /*!< Offset: 0x048 (R/ ) Debug Feature Register           */
__I uint32_t ADR;           /*!< Offset: 0x04C (R/ ) Auxiliary Feature Register         */
__I uint32_t MMFR[4];         /*!< Offset: 0x050 (R/ ) Memory Model Feature Register    */
*/
__I uint32_t ISAR[5];         /*!< Offset: 0x060 (R/ ) Instruction Set Attributes Register */
uint32_t RESERVED0[5];
__IO uint32_t CPACR;          /*!< Offset: 0x088 (R/W) Coprocessor Access Control Register */
*/
} SCB_Type;

```

/* SCB CPUID Register Definitions */

```

#define SCB_CPUID_IMPLEMENTER_Pos    24          /*!< SCB CPUID:  
IMPLEMENTER Position */  
  

#define SCB_CPUID_IMPLEMENTER_Msk    (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)  /*!<  
SCB CPUID: IMPLEMENTER Mask */  
  
  

#define SCB_CPUID_VARIANT_Pos       20          /*!< SCB CPUID: VARIANT Position  
*/  
  

#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos)      /*!< SCB CPUID:  
VARIANT Mask */  
  
  

#define SCB_CPUID_ARCHITECTURE_Pos   16          /*!< SCB CPUID:  
ARCHITECTURE Position */  
  

#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)  /*!< SCB  
CPUID: ARCHITECTURE Mask */  
  
  

#define SCB_CPUID_PARTNO_Pos        4           /*!< SCB CPUID: PARTNO Position  
*/  
  

#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos)     /*!< SCB  
CPUID: PARTNO Mask */  
  
  

#define SCB_CPUID_REVISION_Pos      0           /*!< SCB CPUID: REVISION Position  
*/  
  

#define SCB_CPUID_REVISION_Msk      (0xFUL /*<< SCB_CPUID_REVISION_Pos*/)  /*!< SCB  
CPUID: REVISION Mask */  
  
  

/* SCB Interrupt Control State Register Definitions */  
  

#define SCB_ICSR_NMIPENDSET_Pos    31          /*!< SCB ICSR: NMIPENDSET  
Position */  
  

#define SCB_ICSR_NMIPENDSET_Msk    (1UL << SCB_ICSR_NMIPENDSET_Pos)      /*!< SCB ICSR:  
NMIPENDSET Mask */  
  
  

#define SCB_ICSR_PENDSVSET_Pos     28          /*!< SCB ICSR: PENDSVSET  
Position */

```

#define SCB_ICSR_PENDSVSET_Msk	(1UL << SCB_ICSR_PENDSVSET_Pos)	/*!< SCB ICSR: PENDSVSET Mask */
#define SCB_ICSR_PENDSVCLR_Pos	27	/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */
#define SCB_ICSR_VECTPENDING_Pos	12	/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk	(0x1FFUL << SCB_ICSR_VECTPENDING_Pos)	/*!< SCB ICSR: VECTPENDING Mask */

```

#define SCB_ICSR_RETTOBASE_Pos      11          /*!< SCB ICSR: RETTOBASE
Position */

#define SCB_ICSR_RETTOBASE_Msk     (1UL << SCB_ICSR_RETTOBASE_Pos)    /*!< SCB ICSR:
RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos    0           /*!< SCB ICSR: VECTACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk   (0x1FFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/) /*!< SCB
ICSR: VECTACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */

#define SCB_VTOR_TBLOFF_Pos       7           /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk      (0xFFFFFFFFUL << SCB_VTOR_TBLOFF_Pos) /*!< SCB
VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos     16          /*!< SCB AIRCR: VECTKEY Position */

#define SCB_AIRCR_VECTKEY_Msk    (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos) /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos 16          /*!< SCB AIRCR: VECTKEYSTAT
Position */

#define SCB_AIRCR_VECTKEYSTAT_Msk (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos) /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos   15          /*!< SCB AIRCR: ENDIANESS
Position */

#define SCB_AIRCR_ENDIANESS_Msk  (1UL << SCB_AIRCR_ENDIANESS_Pos) /*!< SCB
AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_PRIGROUP_Pos    8           /*!< SCB AIRCR: PRIGROUP
Position */

```

```

#define SCB_AIRCR_PRIGROUP_Msk      (7UL << SCB_AIRCR_PRIGROUP_Pos)      /*!< SCB
AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos    2                                /*!< SCB AIRCR: SYSRESETREQ
Position */

#define SCB_AIRCR_SYSRESETREQ_Msk    (1UL << SCB_AIRCR_SYSRESETREQ_Pos)      /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos   1                                /*!< SCB AIRCR: VECTCLRACTIVE
Position */

#define SCB_AIRCR_VECTCLRACTIVE_Msk   (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)      /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

#define SCB_AIRCR_VECTRESET_Pos       0                                /*!< SCB AIRCR: VECTRESET
Position */

#define SCB_AIRCR_VECTRESET_Msk       (1UL /*<< SCB_AIRCR_VECTRESET_Pos*/)      /*!< SCB
AIRCR: VECTRESET Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos        4                                /*!< SCB SCR: SEVONPEND
Position */

#define SCB_SCR_SEVONPEND_Msk        (1UL << SCB_SCR_SEVONPEND_Pos)      /*!< SCB SCR:
SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos         2                                /*!< SCB SCR: SLEEPDEEP Position
*/

#define SCB_SCR_SLEEPDEEP_Msk        (1UL << SCB_SCR_SLEEPDEEP_Pos)      /*!< SCB SCR:
SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos       1                                /*!< SCB SCR: SLEEPONEXIT
Position */

#define SCB_SCR_SLEEPONEXIT_Msk       (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR:
SLEEPONEXIT Mask */

```

```

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos      9          /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk     (1UL << SCB_CCR_STKALIGN_Pos)    /*!< SCB CCR:
STKALIGN Mask */

#define SCB_CCR_BFHFMIGN_Pos     8          /*!< SCB CCR: BFHFMIGN
Position */
#define SCB_CCR_BFHFMIGN_Msk   (1UL << SCB_CCR_BFHFMIGN_Pos)    /*!< SCB CCR:
BFHFMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos    4          /*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk  (1UL << SCB_CCR_DIV_0_TRP_Pos)    /*!< SCB CCR:
DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos   3          /*!< SCB CCR: UNALIGN_TRP
Position */
#define SCB_CCR_UNALIGN_TRP_Msk (1UL << SCB_CCR_UNALIGN_TRP_Pos)    /*!< SCB
CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos  1          /*!< SCB CCR: USERSETMPEND
Position */
#define SCB_CCR_USERSETMPEND_Msk (1UL << SCB_CCR_USERSETMPEND_Pos)    /*!< SCB
CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDEN_Pos 0          /*!< SCB CCR:
NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDEN_Msk (1UL /*<< SCB_CCR_NONBASETHRDEN_Pos*/) /*!< SCB CCR:
NONBASETHRDENA Mask */

/* SCB System Handler Control and State Register Definitions */

```

```

#define SCB_SHCSR_USGFAULTENA_Pos    18           /*!< SCB SHCSR:  

USGFAULTENA Position */  
  

#define SCB_SHCSR_USGFAULTENA_Msk   (1UL << SCB_SHCSR_USGFAULTENA_Pos)  /*!< SCB  

SHCSR: USGFAULTENA Mask */  
  

#define SCB_SHCSR_BUSFAULTENA_Pos   17           /*!< SCB SHCSR: BUSFAULTENA  

Position */  
  

#define SCB_SHCSR_BUSFAULTENA_Msk   (1UL << SCB_SHCSR_BUSFAULTENA_Pos)  /*!< SCB  

SHCSR: BUSFAULTENA Mask */  
  

#define SCB_SHCSR_MEMFAULTENA_Pos   16           /*!< SCB SHCSR:  

MEMFAULTENA Position */  
  

#define SCB_SHCSR_MEMFAULTENA_Msk   (1UL << SCB_SHCSR_MEMFAULTENA_Pos)  /*!<  

SCB SHCSR: MEMFAULTENA Mask */  
  

#define SCB_SHCSR_SVCALLPENDED_Pos  15           /*!< SCB SHCSR:  

SVCALLPENDED Position */  
  

#define SCB_SHCSR_SVCALLPENDED_Msk  (1UL << SCB_SHCSR_SVCALLPENDED_Pos)  /*!< SCB  

SHCSR: SVCALLPENDED Mask */  
  

#define SCB_SHCSR_BUSFAULTPENDED_Pos 14           /*!< SCB SHCSR:  

BUSFAULTPENDED Position */  
  

#define SCB_SHCSR_BUSFAULTPENDED_Msk (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)  /*!<  

SCB SHCSR: BUSFAULTPENDED Mask */  
  

#define SCB_SHCSR_MEMFAULTPENDED_Pos 13           /*!< SCB SHCSR:  

MEMFAULTPENDED Position */  
  

#define SCB_SHCSR_MEMFAULTPENDED_Msk (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)  

/*!< SCB SHCSR: MEMFAULTPENDED Mask */  
  

#define SCB_SHCSR_USGFAULTPENDED_Pos 12           /*!< SCB SHCSR:  

USGFAULTPENDED Position */  
  

#define SCB_SHCSR_USGFAULTPENDED_Msk (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)  /*!<  

SCB SHCSR: USGFAULTPENDED Mask */

```

#define SCB_SHCSR_SYSTICKACT_Pos	11	/*!< SCB SHCSR: SYSTICKACT Position */
#define SCB_SHCSR_SYSTICKACT_Msk	(1UL << SCB_SHCSR_SYSTICKACT_Pos)	/*!< SCB SHCSR: SYSTICKACT Mask */
#define SCB_SHCSR_PENDSVACT_Pos	10	/*!< SCB SHCSR: PENDSVACT Position */
#define SCB_SHCSR_PENDSVACT_Msk	(1UL << SCB_SHCSR_PENDSVACT_Pos)	/*!< SCB SHCSR: PENDSVACT Mask */
#define SCB_SHCSR_MONITORACT_Pos	8	/*!< SCB SHCSR: MONITORACT Position */
#define SCB_SHCSR_MONITORACT_Msk	(1UL << SCB_SHCSR_MONITORACT_Pos)	/*!< SCB SHCSR: MONITORACT Mask */
#define SCB_SHCSR_SVCALLACT_Pos	7	/*!< SCB SHCSR: SVCALLACT Position */
#define SCB_SHCSR_SVCALLACT_Msk	(1UL << SCB_SHCSR_SVCALLACT_Pos)	/*!< SCB SHCSR: SVCALLACT Mask */
#define SCB_SHCSR_USGFAULTACT_Pos	3	/*!< SCB SHCSR: USGFAULTACT Position */
#define SCB_SHCSR_USGFAULTACT_Msk	(1UL << SCB_SHCSR_USGFAULTACT_Pos)	/*!< SCB SHCSR: USGFAULTACT Mask */
#define SCB_SHCSR_BUSFAULTACT_Pos	1	/*!< SCB SHCSR: BUSFAULTACT Position */
#define SCB_SHCSR_BUSFAULTACT_Msk	(1UL << SCB_SHCSR_BUSFAULTACT_Pos)	/*!< SCB SHCSR: BUSFAULTACT Mask */
#define SCB_SHCSR_MEMFAULTACT_Pos	0	/*!< SCB SHCSR: MEMFAULTACT Position */

```

#define SCB_SHCSR_MEMFAULTACT_Msk      (1UL /*<< SCB_SHCSR_MEMFAULTACT_Pos*/) /*!<
SCB SHCSR: MEMFAULTACT Mask */

/* SCB Configurable Fault Status Registers Definitions */

#define SCB_CFSR_USGFAULTSR_Pos       16                                /*!< SCB CFSR: Usage Fault
Status Register Position */

#define SCB_CFSR_USGFAULTSR_Msk      (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos) /*!< SCB
CFSR: Usage Fault Status Register Mask */

#define SCB_CFSR_BUSFAULTSR_Pos       8                                 /*!< SCB CFSR: Bus Fault Status
Register Position */

#define SCB_CFSR_BUSFAULTSR_Msk      (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos) /*!< SCB
CFSR: Bus Fault Status Register Mask */

#define SCB_CFSR_MEMFAULTSR_Pos       0                                /*!< SCB CFSR: Memory Manage
Fault Status Register Position */

#define SCB_CFSR_MEMFAULTSR_Msk      (0xFFUL /*<< SCB_CFSR_MEMFAULTSR_Pos*/) /*!<
SCB CFSR: Memory Manage Fault Status Register Mask */

/* SCB Hard Fault Status Registers Definitions */

#define SCB_HFSR_DEBUGEVT_Pos        31                                /*!< SCB HFSR: DEBUGEVT
Position */

#define SCB_HFSR_DEBUGEVT_Msk       (1UL << SCB_HFSR_DEBUGEVT_Pos) /*!< SCB HFSR:
DEBUGEVT Mask */

#define SCB_HFSR_FORCED_Pos         30                                /*!< SCB HFSR: FORCED Position */

#define SCB_HFSR_FORCED_Msk        (1UL << SCB_HFSR_FORCED_Pos) /*!< SCB HFSR:
FORCED Mask */

#define SCB_HFSR_VECTTBL_Pos        1                                 /*!< SCB HFSR: VECTTBL Position */

#define SCB_HFSR_VECTTBL_Msk       (1UL << SCB_HFSR_VECTTBL_Pos) /*!< SCB HFSR:
VECTTBL Mask */

```

```

/* SCB Debug Fault Status Register Definitions */

#define SCB_DFSR_EXTERNAL_Pos      4                  /*!< SCB DFSR: EXTERNAL Position */
/*/

#define SCB_DFSR_EXTERNAL_Msk     (1UL << SCB_DFSR_EXTERNAL_Pos)    /*!< SCB DFSR:
EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos       3                  /*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk       (1UL << SCB_DFSR_VCATCH_Pos)    /*!< SCB DFSR:
VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos      2                  /*!< SCB DFSR: DWTTRAP Position */
/*/

#define SCB_DFSR_DWTTRAP_Msk     (1UL << SCB_DFSR_DWTTRAP_Pos)    /*!< SCB DFSR:
DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos         1                  /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk         (1UL << SCB_DFSR_BKPT_Pos)      /*!< SCB DFSR: BKPT
Mask */

#define SCB_DFSR_HALTED_Pos       0                  /*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk       (1UL /*<< SCB_DFSR_HALTED_Pos*/) /*!< SCB DFSR:
HALTED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
\brief Type definitions for the System Control and ID Register not in the SCB
@{
*/

```

```

/** \brief Structure type to access the System Control and ID Register not in the SCB.
 */
typedef struct
{
    uint32_t RESERVED0[1];

    __I uint32_t ICTR;           /*!< Offset: 0x004 (R/ ) Interrupt Controller Type Register */
    __IO uint32_t ACTLR;         /*!< Offset: 0x008 (R/W) Auxiliary Control Register */

} SCnSCB_Type;

/* Interrupt Controller Type Register Definitions */

#define SCnSCB_ICTR_INTLINESNUM_Pos      0          /*!< ICTR: INTLINESNUM Position */
/* */

#define SCnSCB_ICTR_INTLINESNUM_Msk     (0xFUL /*<< SCnSCB_ICTR_INTLINESNUM_Pos*/) /*!<
ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define SCnSCB_ACTLR_DISOOPF_Pos       9          /*!< ACTLR: DISOOPF Position */
#define SCnSCB_ACTLR_DISOOPF_Msk       (1UL << SCnSCB_ACTLR_DISOOPF_Pos) /*!< ACTLR:
DISOOPF Mask */

#define SCnSCB_ACTLR_DISFPCA_Pos       8          /*!< ACTLR: DISFPCA Position */
#define SCnSCB_ACTLR_DISFPCA_Msk       (1UL << SCnSCB_ACTLR_DISFPCA_Pos) /*!< ACTLR:
DISFPCA Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos       2          /*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk       (1UL << SCnSCB_ACTLR_DISFOLD_Pos) /*!< ACTLR:
DISFOLD Mask */

#define SCnSCB_ACTLR_DISDEFWBUF_Pos    1          /*!< ACTLR: DISDEFWBUF
Position */

```

```

#define SCnSCB_ACTLR_DISDEFWBUF_Msk      (1UL << SCnSCB_ACTLR_DISDEFWBUF_Pos)      /*!<
ACTLR: DISDEFWBUF Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos      0                                /*!< ACTLR: DISMCYCINT
Position */

#define SCnSCB_ACTLR_DISMCYCINT_Msk      (1UL /*<< SCnSCB_ACTLR_DISMCYCINT_Pos*/)  /*!<
ACTLR: DISMCYCINT Mask */

/*}@} end of group CMSIS_SCnotSCB */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SysTick System Tick Timer (SysTick)
\brief Type definitions for the System Timer Registers.

@{

*/

```

/** \brief Structure type to access the System Timer (SysTick).

```

*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I uint32_t CALIB;          /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
} SysTick_Type;
```

/* SysTick Control / Status Register Definitions */

```

#define SysTick_CTRL_COUNTFLAG_Pos     16                                /*!< SysTick CTRL: COUNTFLAG
Position */

```

```

#define SysTick_CTRL_COUNTFLAG_Msk      (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick
CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos      2                                /*!< SysTick CTRL: CLKSOURCE
Position */

#define SysTick_CTRL_CLKSOURCE_Msk      (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick
CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos       1                                /*!< SysTick CTRL: TICKINT Position */
*/

#define SysTick_CTRL_TICKINT_Msk      (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL:
TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos        0                                /*!< SysTick CTRL: ENABLE Position */
*/

#define SysTick_CTRL_ENABLE_Msk      (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)      /*!< SysTick
CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos        0                                /*!< SysTick LOAD: RELOAD
Position */

#define SysTick_LOAD_RELOAD_Msk      (0xFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/)  /*!<
SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos        0                                /*!< SysTick VAL: CURRENT
Position */

#define SysTick_VAL_CURRENT_Msk      (0xFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/)  /*!<
SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos       31                               /*!< SysTick CALIB: NOREF Position
*/

```

```

#define SysTick_CALIB_NOREF_Msk      (1UL << SysTick_CALIB_NOREF_Pos)      /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos       30                                /*!< SysTick CALIB: SKEW Position
*/
#define SysTick_CALIB_SKEW_Msk      (1UL << SysTick_CALIB_SKEW_Pos)      /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos     0                                 /*!< SysTick CALIB: TENMS Position
*/
#define SysTick_CALIB_TENMS_Msk    (0xFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_ITM  Instrumentation Trace Macrocell (ITM)
\brief   Type definitions for the Instrumentation Trace Macrocell (ITM)

@{
*/

```

```
/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
```

```
/*
typedef struct
{
    __O union
    {
        __O uint8_t  u8;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit      */
        __O uint16_t u16;         /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit      */
    }
}
```

```

    __O uint32_t u32;           /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit      */
} PORT [32];           /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers      */

    uint32_t RESERVED0[864];

__IO uint32_t TER;           /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register      */
    uint32_t RESERVED1[15];

__IO uint32_t TPR;           /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register      */
    uint32_t RESERVED2[15];

__IO uint32_t TCR;           /*!< Offset: 0xE80 (R/W) ITM Trace Control Register      */
    uint32_t RESERVED3[29];

__O uint32_t IWR;           /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register      */
__I uint32_t IRR;           /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register      */
__IO uint32_t IMCR;           /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
    uint32_t RESERVED4[43];

__O uint32_t LAR;           /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register      */
__I uint32_t LSR;           /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register      */
    uint32_t RESERVED5[6];

__I uint32_t PID4;           /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
__I uint32_t PID5;           /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
__I uint32_t PID6;           /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
__I uint32_t PID7;           /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
__I uint32_t PID0;           /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
__I uint32_t PID1;           /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
__I uint32_t PID2;           /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
__I uint32_t PID3;           /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
__I uint32_t CID0;           /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
__I uint32_t CID1;           /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
__I uint32_t CID2;           /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */
__I uint32_t CID3;           /*!< Offset: 0xFFC (R/ ) ITM Component Identification Register #3 */

} ITM_Type;

```

```

/* ITM Trace Privilege Register Definitions */

#define ITM_TPR_PRIVMASK_Pos      0           /*!< ITM TPR: PRIVMASK Position */
/* */

#define ITM_TPR_PRIVMASK_Msk     (0xFUL /*<< ITM_TPR_PRIVMASK_Pos*/)    /*!< ITM TPR: PRIVMASK Mask */

/* ITM Trace Control Register Definitions */

#define ITM_TCR_BUSY_Pos        23          /*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk        (1UL << ITM_TCR_BUSY_Pos)    /*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos   16          /*!< ITM TCR: ATBID Position */
#define ITM_TCR_TraceBusID_Msk  (0x7FUL << ITM_TCR_TraceBusID_Pos)  /*!< ITM TCR: ATBID Mask */

#define ITM_TCR_GTSFREQ_Pos     10          /*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk    (3UL << ITM_TCR_GTSFREQ_Pos)    /*!< ITM TCR: Global timestamp frequency Mask */

#define ITM_TCR_TSPrescale_Pos   8           /*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk  (3UL << ITM_TCR_TSPrescale_Pos)  /*!< ITM TCR: TSPrescale Mask */

#define ITM_TCR_SWOENA_Pos      4            /*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk     (1UL << ITM_TCR_SWOENA_Pos)    /*!< ITM TCR: SWOENA Mask */

#define ITM_TCR_DWTENA_Pos      3            /*!< ITM TCR: DWTENA Position */
#define ITM_TCR_DWTENA_Msk     (1UL << ITM_TCR_DWTENA_Pos)    /*!< ITM TCR: DWTENA Mask */

```

```
#define ITM_TCR_SYNCENA_Pos      2          /*!< ITM TCR: SYNCENA Position */  
#define ITM_TCR_SYNCENA_Msk      (1UL << ITM_TCR_SYNCENA_Pos)      /*!< ITM TCR:  
SYNCENA Mask */
```

```
#define ITM_TCR_TSENA_Pos       1          /*!< ITM TCR: TSENA Position */  
#define ITM_TCR_TSENA_Msk       (1UL << ITM_TCR_TSENA_Pos)      /*!< ITM TCR: TSENA  
Mask */
```

```
#define ITM_TCR_ITMENA_Pos      0          /*!< ITM TCR: ITM Enable bit  
Position */  
#define ITM_TCR_ITMENA_Msk      (1UL /*<< ITM_TCR_ITMENA_Pos*/)      /*!< ITM TCR:  
ITM Enable bit Mask */
```

/ ITM Integration Write Register Definitions */*

```
#define ITM_IWR_ATVALIDDM_Pos    0          /*!< ITM IWR: ATVALIDDM Position  
*/  
#define ITM_IWR_ATVALIDDM_Msk    (1UL /*<< ITM_IWR_ATVALIDDM_Pos*/)      /*!< ITM IWR:  
ATVALIDDM Mask */
```

/ ITM Integration Read Register Definitions */*

```
#define ITMIRR_ATREADYM_Pos     0          /*!< ITM IRR: ATREADYM Position  
*/  
#define ITMIRR_ATREADYM_Msk     (1UL /*<< ITMIRR_ATREADYM_Pos*/)      /*!< ITM IRR:  
ATREADYM Mask */
```

/ ITM Integration Mode Control Register Definitions */*

```
#define ITMIMCR_INTEGRATION_Pos  0          /*!< ITM IMCR: INTEGRATION  
Position */  
#define ITMIMCR_INTEGRATION_Msk  (1UL /*<< ITMIMCR_INTEGRATION_Pos*/)      /*!< ITM  
IMCR: INTEGRATION Mask */
```

```

/* ITM Lock Status Register Definitions */

#define ITM_LSR_ByteAcc_Pos      2          /*!< ITM LSR: ByteAcc Position */
#define ITM_LSR_ByteAcc_Msk      (1UL << ITM_LSR_ByteAcc_Pos)      /*!< ITM LSR: ByteAcc
Mask */

#define ITM_LSR_Access_Pos      1          /*!< ITM LSR: Access Position */
#define ITM_LSR_Access_Msk      (1UL << ITM_LSR_Access_Pos)      /*!< ITM LSR: Access
Mask */

#define ITM_LSR_Present_Pos     0          /*!< ITM LSR: Present Position */
#define ITM_LSR_Present_Msk     (1UL /*<< ITM_LSR_Present_Pos*/)      /*!< ITM LSR:
Present Mask */

/*@}*/ /* end of group CMSIS_ITM */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
\brief Type definitions for the Data Watchpoint and Trace (DWT)

@{

*/

```

/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).

```

typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) Control Register */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle Count Register */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI Count Register */
    __IO uint32_t EXCCNT;        /*!< Offset: 0x00C (R/W) Exception Overhead Count Register */

```

```

__IO uint32_t SLEEPCNT;           /*!< Offset: 0x010 (R/W) Sleep Count Register      */
__IO uint32_t LSUCNT;            /*!< Offset: 0x014 (R/W) LSU Count Register        */
__IO uint32_t FOLDCNT;           /*!< Offset: 0x018 (R/W) Folded-instruction Count Register   */
__I uint32_t PCSR;              /*!< Offset: 0x01C (R/ ) Program Counter Sample Register */
__IO uint32_t COMPO;             /*!< Offset: 0x020 (R/W) Comparator Register 0       */
__IO uint32_t MASK0;             /*!< Offset: 0x024 (R/W) Mask Register 0           */
__IO uint32_t FUNCTION0;         /*!< Offset: 0x028 (R/W) Function Register 0        */
    uint32_t RESERVED0[1];
__IO uint32_t COMP1;             /*!< Offset: 0x030 (R/W) Comparator Register 1       */
__IO uint32_t MASK1;             /*!< Offset: 0x034 (R/W) Mask Register 1           */
__IO uint32_t FUNCTION1;         /*!< Offset: 0x038 (R/W) Function Register 1        */
    uint32_t RESERVED1[1];
__IO uint32_t COMP2;             /*!< Offset: 0x040 (R/W) Comparator Register 2       */
__IO uint32_t MASK2;             /*!< Offset: 0x044 (R/W) Mask Register 2           */
__IO uint32_t FUNCTION2;         /*!< Offset: 0x048 (R/W) Function Register 2        */
    uint32_t RESERVED2[1];
__IO uint32_t COMP3;             /*!< Offset: 0x050 (R/W) Comparator Register 3       */
__IO uint32_t MASK3;             /*!< Offset: 0x054 (R/W) Mask Register 3           */
__IO uint32_t FUNCTION3;         /*!< Offset: 0x058 (R/W) Function Register 3        */

} DWT_Type;

```

```

/* DWT Control Register Definitions */

#define DWT_CTRL_NUMCOMP_Pos      28          /*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk      (0xFUL << DWT_CTRL_NUMCOMP_Pos) /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos     27          /*!< DWT CTRL: NOTRCPKT Position */

```

#define DWT_CTRL_NOTRCPKT_Msk	(0x1UL << DWT_CTRL_NOTRCPKT_Pos)	/*!< DWT
CTRL: NOTRCPKT Mask */		
#define DWT_CTRL_NOEXTTRIG_Pos	26	/*!< DWT CTRL: NOEXTTRIG
Position */		
#define DWT_CTRL_NOEXTTRIG_Msk	(0x1UL << DWT_CTRL_NOEXTTRIG_Pos)	/*!< DWT
CTRL: NOEXTTRIG Mask */		
#define DWT_CTRL_NOCYCCNT_Pos	25	/*!< DWT CTRL: NOCYCCNT
Position */		
#define DWT_CTRL_NOCYCCNT_Msk	(0x1UL << DWT_CTRL_NOCYCCNT_Pos)	/*!< DWT
CTRL: NOCYCCNT Mask */		
#define DWT_CTRL_NOPRFCNT_Pos	24	/*!< DWT CTRL: NOPRFCNT
Position */		
#define DWT_CTRL_NOPRFCNT_Msk	(0x1UL << DWT_CTRL_NOPRFCNT_Pos)	/*!< DWT
CTRL: NOPRFCNT Mask */		
#define DWT_CTRL_CYCEVTENA_Pos	22	/*!< DWT CTRL: CYCEVTENA
Position */		
#define DWT_CTRL_CYCEVTENA_Msk	(0x1UL << DWT_CTRL_CYCEVTENA_Pos)	/*!< DWT
CTRL: CYCEVTENA Mask */		
#define DWT_CTRL_FOLDEVTENA_Pos	21	/*!< DWT CTRL: FOLDEVTENA
Position */		
#define DWT_CTRL_FOLDEVTENA_Msk	(0x1UL << DWT_CTRL_FOLDEVTENA_Pos)	/*!< DWT
CTRL: FOLDEVTENA Mask */		
#define DWT_CTRL_LSUEVTENA_Pos	20	/*!< DWT CTRL: LSUEVTENA
Position */		
#define DWT_CTRL_LSUEVTENA_Msk	(0x1UL << DWT_CTRL_LSUEVTENA_Pos)	/*!< DWT
CTRL: LSUEVTENA Mask */		

#define DWT_CTRL_SLEEPEVTENA_Pos	19	/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk	(0x1UL << DWT_CTRL_SLEEPEVTENA_Pos)	/*!< DWT CTRL: SLEEPEVTENA Mask */
#define DWT_CTRL_EXCEVTENA_Pos	18	/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk	(0x1UL << DWT_CTRL_EXCEVTENA_Pos)	/*!< DWT CTRL: EXCEVTENA Mask */
#define DWT_CTRL_CPIEVTEA_Pos	17	/*!< DWT CTRL: CPIEVTEA Position */
#define DWT_CTRL_CPIEVTEA_Msk	(0x1UL << DWT_CTRL_CPIEVTEA_Pos)	/*!< DWT CTRL: CPIEVTEA Mask */
#define DWT_CTRL_EXCTRCENA_Pos	16	/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk	(0x1UL << DWT_CTRL_EXCTRCENA_Pos)	/*!< DWT CTRL: EXCTRCENA Mask */
#define DWT_CTRL_PCSAMPLENA_Pos	12	/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk	(0x1UL << DWT_CTRL_PCSAMPLENA_Pos)	/*!< DWT CTRL: PCSAMPLENA Mask */
#define DWT_CTRL_SYNCTAP_Pos	10	/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk	(0x3UL << DWT_CTRL_SYNCTAP_Pos)	/*!< DWT CTRL: SYNCTAP Mask */
#define DWT_CTRL_CYCTAP_Pos	9	/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk	(0x1UL << DWT_CTRL_CYCTAP_Pos)	/*!< DWT CTRL: CYCTAP Mask */

```

#define DWT_CTRL_POSTINIT_Pos      5          /*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk      (0xFUL << DWT_CTRL_POSTINIT_Pos)    /*!< DWT CTRL:
POSTINIT Mask */

#define DWT_CTRL_POSTPRESET_Pos    1          /*!< DWT CTRL: POSTPRESET
Position */
#define DWT_CTRL_POSTPRESET_Msk    (0xFUL << DWT_CTRL_POSTPRESET_Pos)    /*!< DWT
CTRL: POSTPRESET Mask */

#define DWT_CTRL_CYCCNTENA_Pos     0          /*!< DWT CTRL: CYCCNTENA
Position */
#define DWT_CTRL_CYCCNTENA_Msk    (0x1UL /*<< DWT_CTRL_CYCCNTENA_Pos*/) /*!< DWT
CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */

#define DWT_CPICNT_CPICNT_Pos     0          /*!< DWT CPICNT: CPICNT Position */
#define DWT_CPICNT_CPICNT_Msk     (0xFFUL /*<< DWT_CPICNT_CPICNT_Pos*/) /*!< DWT
CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */

#define DWT_EXCCNT_EXCCNT_Pos     0          /*!< DWT EXCCNT: EXCCNT
Position */
#define DWT_EXCCNT_EXCCNT_Msk     (0xFFUL /*<< DWT_EXCCNT_EXCCNT_Pos*/) /*!< DWT
EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */

#define DWT_SLEEPcnt_SLEEPcnt_Pos 0          /*!< DWT SLEEPcnt: SLEEPcnt
Position */
#define DWT_SLEEPcnt_SLEEPcnt_Msk (0xFFUL /*<< DWT_SLEEPcnt_SLEEPcnt_Pos*/) /*!<
DWT SLEEPcnt: SLEEPcnt Mask */

```

```

/* DWT LSU Count Register Definitions */

#define DWT_LSUCNT_LSUCNT_Pos      0           /*!< DWT LSUCNT: LSUCNT Position
*/
#define DWT_LSUCNT_LSUCNT_Msk     (0xFFUL /*<< DWT_LSUCNT_LSUCNT_Pos*/) /*!< DWT
LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */

#define DWT_FOLDCNT_FOLDCNT_Pos    0           /*!< DWT FOLDCNT: FOLDCNT
Position */
#define DWT_FOLDCNT_FOLDCNT_Msk   (0xFFUL /*<< DWT_FOLDCNT_FOLDCNT_Pos*/) /*!<
DWT FOLDCNT: FOLDCNT Mask */

/* DWT Comparator Mask Register Definitions */

#define DWT_MASK_MASK_Pos         0           /*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk        (0x1FUL /*<< DWT_MASK_MASK_Pos*/) /*!< DWT
MASK: MASK Mask */

/* DWT Comparator Function Register Definitions */

#define DWT_FUNCTION_MATCHED_Pos   24          /*!< DWT FUNCTION:
MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk  (0x1UL << DWT_FUNCTION_MATCHED_Pos) /*!<
DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16          /*!< DWT FUNCTION:
DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL << DWT_FUNCTION_DATAVADDR1_Pos) /*!<
DWT FUNCTION: DATAVADDR1 Mask */

#define DWT_FUNCTION_DATAVADDR0_Pos 12          /*!< DWT FUNCTION:
DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk (0xFUL << DWT_FUNCTION_DATAVADDR0_Pos) /*!<
DWT FUNCTION: DATAVADDR0 Mask */

```

```

#define DWT_FUNCTION_DATAVSIZE_Pos    10           /*!< DWT FUNCTION:  

DATAVSIZE Position */  
  

#define DWT_FUNCTION_DATAVSIZE_Msk   (0x3UL << DWT_FUNCTION_DATAVSIZE_Pos)  /*!<  

DWT FUNCTION: DATAVSIZE Mask */  
  

#define DWT_FUNCTION_LNK1ENA_Pos     9            /*!< DWT FUNCTION: LNK1ENA  

Position */  
  

#define DWT_FUNCTION_LNK1ENA_Msk    (0x1UL << DWT_FUNCTION_LNK1ENA_Pos)  /*!<  

DWT FUNCTION: LNK1ENA Mask */  
  

#define DWT_FUNCTION_DATAVMATCH_Pos  8            /*!< DWT FUNCTION:  

DATAVMATCH Position */  
  

#define DWT_FUNCTION_DATAVMATCH_Msk  (0x1UL << DWT_FUNCTION_DATAVMATCH_Pos)  

/*!< DWT FUNCTION: DATAVMATCH Mask */  
  

#define DWT_FUNCTION_CYCMATCH_Pos    7            /*!< DWT FUNCTION:  

CYCMATCH Position */  
  

#define DWT_FUNCTION_CYCMATCH_Msk   (0x1UL << DWT_FUNCTION_CYCMATCH_Pos)  /*!<  

DWT FUNCTION: CYCMATCH Mask */  
  

#define DWT_FUNCTION_EMITRANGE_Pos   5            /*!< DWT FUNCTION:  

EMITRANGE Position */  
  

#define DWT_FUNCTION_EMITRANGE_Msk  (0x1UL << DWT_FUNCTION_EMITRANGE_Pos)  /*!<  

DWT FUNCTION: EMITRANGE Mask */  
  

#define DWT_FUNCTION_FUNCTION_Pos    0            /*!< DWT FUNCTION:  

FUNCTION Position */  
  

#define DWT_FUNCTION_FUNCTION_Msk   (0xFUL /*<< DWT_FUNCTION_FUNCTION_Pos*/)  /*!<  

DWT FUNCTION: FUNCTION Mask */  
  

/*@}*/ /* end of group CMSIS_DWT */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_TPI Trace Port Interface (TPI)
 \brief Type definitions for the Trace Port Interface (TPI)

 @{
 */

/** \brief Structure type to access the Trace Port Interface Register (TPI).
 */
typedef struct
{
    __IO uint32_t SSPSR;           /*!< Offset: 0x000 (R/ ) Supported Parallel Port Size Register */
    __IO uint32_t CSPSR;          /*!< Offset: 0x004 (R/W) Current Parallel Port Size Register */
    uint32_t RESERVED0[2];
    __IO uint32_t ACPR;           /*!< Offset: 0x010 (R/W) Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];
    __IO uint32_t SPPR;           /*!< Offset: 0x0F0 (R/W) Selected Pin Protocol Register */
    uint32_t RESERVED2[131];
    __I uint32_t FFSR;            /*!< Offset: 0x300 (R/ ) Formatter and Flush Status Register */
    __IO uint32_t FFCR;           /*!< Offset: 0x304 (R/W) Formatter and Flush Control Register */
    __I uint32_t FSCR;            /*!< Offset: 0x308 (R/ ) Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];
    __I uint32_t TRIGGER;         /*!< Offset: 0xEE8 (R/ ) TRIGGER */
    __I uint32_t FIFO0;           /*!< Offset: 0xEEC (R/ ) Integration ETM Data */
    __I uint32_t ITATBCTR2;       /*!< Offset: 0xEF0 (R/ ) ITATBCTR2 */
    uint32_t RESERVED4[1];
    __I uint32_t ITATBCTR0;       /*!< Offset: 0xEF8 (R/ ) ITATBCTR0 */
    __I uint32_t FIFO1;           /*!< Offset: 0xEFC (R/ ) Integration ITM Data */
    __IO uint32_t ITCTRL;          /*!< Offset: 0xF00 (R/W) Integration Mode Control */
    uint32_t RESERVED5[39];
}

```

```

__IO uint32_t CLAIMSET;           /*!< Offset: 0xFA0 (R/W) Claim tag set */
__IO uint32_t CLAIMCLR;          /*!< Offset: 0xFA4 (R/W) Claim tag clear */
uint32_t RESERVED7[8];

__I uint32_t DEVID;             /*!< Offset: 0xFC8 (R/ ) TPIU_DEVID */
__I uint32_t DEVTYPE;           /*!< Offset: 0xFCC (R/ ) TPIU_DEVTYPE */

} TPI_Type;

/* TPI Asynchronous Clock Prescaler Register Definitions */

#define TPI_ACPR_PRESCALER_Pos      0           /*!< TPI ACPR: PRESCALER Position */
#define TPI_ACPR_PRESCALER_Msk      (0xFFFFUL /*<< TPI_ACPR_PRESCALER_Pos*/) /*!< TPI
ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */

#define TPI_SPPR_TXMODE_Pos        0           /*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk        (0x3UL /*<< TPI_SPPR_TXMODE_Pos*/) /*!< TPI SPPR:
TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */

#define TPI_FFSR_FtNonStop_Pos     3           /*!< TPI FFSR: FtNonStop Position */
#define TPI_FFSR_FtNonStop_Msk     (0x1UL << TPI_FFSR_FtNonStop_Pos) /*!< TPI FFSR:
FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos     2           /*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk     (0x1UL << TPI_FFSR_TCPresent_Pos) /*!< TPI FFSR:
TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos     1           /*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk     (0x1UL << TPI_FFSR_FtStopped_Pos) /*!< TPI FFSR:
FtStopped Mask */

```

```

#define TPI_FFSR_FInProg_Pos      0           /*!< TPI FFSR: FInProg Position */
#define TPI_FFSR_FInProg_Msk     (0x1UL /*<< TPI_FFSR_FInProg_Pos*/)    /*!< TPI FFSR:
FInProg Mask */

/* TPI Formatter and Flush Control Register Definitions */

#define TPI_FFCR_TrigIn_Pos      8           /*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk     (0x1UL << TPI_FFCR_TrigIn_Pos)    /*!< TPI FFCR:
TrigIn Mask */

#define TPI_FFCR_EnFCont_Pos      1           /*!< TPI FFCR: EnFCont Position */
#define TPI_FFCR_EnFCont_Msk     (0x1UL << TPI_FFCR_EnFCont_Pos)    /*!< TPI FFCR:
EnFCont Mask */

/* TPI TRIGGER Register Definitions */

#define TPI_TRIGGER_TRIGGER_Pos      0           /*!< TPI TRIGGER: TRIGGER Position */
#define TPI_TRIGGER_TRIGGER_Msk   (0x1UL /*<< TPI_TRIGGER_TRIGGER_Pos*/)  /*!< TPI
TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */

#define TPI_FIFO0_ITM_ATVALID_Pos      29          /*!< TPI FIFO0: ITM_ATVALID
Position */
#define TPI_FIFO0_ITM_ATVALID_Msk   (0x3UL << TPI_FIFO0_ITM_ATVALID_Pos)  /*!< TPI FIFO0:
ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos      27          /*!< TPI FIFO0: ITM_bytecount
Position */
#define TPI_FIFO0_ITM_bytecount_Msk   (0x3UL << TPI_FIFO0_ITM_bytecount_Pos) /*!< TPI
FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos      26          /*!< TPI FIFO0: ETM_ATVALID
Position */

```

```

#define TPI_FIFO0_ETM_ATVALID_Msk      (0x3UL << TPI_FIFO0_ETM_ATVALID_Pos) /*!< TPI
FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos    24                      /*!< TPI FIFO0: ETM_bytecount
Position */

#define TPI_FIFO0_ETM_bytecount_Msk   (0x3UL << TPI_FIFO0_ETM_bytecount_Pos) /*!< TPI
FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos           16                     /*!< TPI FIFO0: ETM2 Position */

#define TPI_FIFO0_ETM2_Msk          (0xFFUL << TPI_FIFO0_ETM2_Pos) /*!< TPI FIFO0: ETM2
Mask */

#define TPI_FIFO0_ETM1_Pos           8                      /*!< TPI FIFO0: ETM1 Position */

#define TPI_FIFO0_ETM1_Msk          (0xFFUL << TPI_FIFO0_ETM1_Pos) /*!< TPI FIFO0: ETM1
Mask */

#define TPI_FIFO0_ETM0_Pos           0                      /*!< TPI FIFO0: ETM0 Position */

#define TPI_FIFO0_ETM0_Msk          (0xFFUL /*<< TPI_FIFO0_ETM0_Pos*/) /*!< TPI FIFO0:
ETM0 Mask */

/* TPI ITATBCTR2 Register Definitions */

#define TPI_ITATBCTR2_ATREADY_Pos    0                      /*!< TPI ITATBCTR2: ATREADY
Position */

#define TPI_ITATBCTR2_ATREADY_Msk   (0x1UL /*<< TPI_ITATBCTR2_ATREADY_Pos*/) /*!< TPI
ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */

#define TPI_FIFO1_ITM_ATVALID_Pos    29                     /*!< TPI FIFO1: ITM_ATVALID
Position */

#define TPI_FIFO1_ITM_ATVALID_Msk   (0x3UL << TPI_FIFO1_ITM_ATVALID_Pos) /*!< TPI FIFO1:
ITM_ATVALID Mask */

```

```

#define TPI_FIFO1_ITM_bytecount_Pos    27          /*!< TPI FIFO1: ITM_bytecount
Position */

#define TPI_FIFO1_ITM_bytecount_Msk   (0x3UL << TPI_FIFO1_ITM_bytecount_Pos) /*!< TPI
FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos    26          /*!< TPI FIFO1: ETM_ATVALID
Position */

#define TPI_FIFO1_ETM_ATVALID_Msk   (0x3UL << TPI_FIFO1_ETM_ATVALID_Pos) /*!< TPI
FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos    24          /*!< TPI FIFO1: ETM_bytecount
Position */

#define TPI_FIFO1_ETM_bytecount_Msk   (0x3UL << TPI_FIFO1_ETM_bytecount_Pos) /*!< TPI
FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos           16          /*!< TPI FIFO1: ITM2 Position */

#define TPI_FIFO1_ITM2_Msk           (0xFFUL << TPI_FIFO1_ITM2_Pos) /*!< TPI FIFO1: ITM2
Mask */

#define TPI_FIFO1_ITM1_Pos           8           /*!< TPI FIFO1: ITM1 Position */

#define TPI_FIFO1_ITM1_Msk           (0xFFUL << TPI_FIFO1_ITM1_Pos) /*!< TPI FIFO1: ITM1
Mask */

#define TPI_FIFO1_ITM0_Pos           0           /*!< TPI FIFO1: ITM0 Position */

#define TPI_FIFO1_ITM0_Msk           (0xFFUL /*<< TPI_FIFO1_ITM0_Pos*/) /*!< TPI FIFO1: ITM0
Mask */

/* TPI ITATBCTRO Register Definitions */

#define TPI_ITATBCTRO_ATREADY_Pos    0          /*!< TPI ITATBCTRO: ATREADY
Position */

#define TPI_ITATBCTRO_ATREADY_Msk   (0x1UL /*<< TPI_ITATBCTRO_ATREADY_Pos*/) /*!< TPI
ITATBCTRO: ATREADY Mask */

```

```

/* TPI Integration Mode Control Register Definitions */

#define TPI_ITCTRL_Mode_Pos      0           /*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk      (0x1UL /*<< TPI_ITCTRL_Mode_Pos*/)    /*!< TPI ITCTRL: Mode Mask */

/* TPI DEVID Register Definitions */

#define TPI_DEVID_NRZVALID_Pos   11          /*!< TPI DEVID: NRZVALID Position */
#define TPI_DEVID_NRZVALID_Msk   (0x1UL << TPI_DEVID_NRZVALID_Pos)    /*!< TPI DEVID: NRZVALID Mask */

#define TPI_DEVID_MANCVALID_Pos  10          /*!< TPI DEVID: MANCVALID Position */
#define TPI_DEVID_MANCVALID_Msk  (0x1UL << TPI_DEVID_MANCVALID_Pos)    /*!< TPI DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALIDID_Pos 9           /*!< TPI DEVID: PTINVALIDID Position */
#define TPI_DEVID_PTINVALIDID_Msk (0x1UL << TPI_DEVID_PTINVALIDID_Pos)    /*!< TPI DEVID: PTINVALIDID Mask */

#define TPI_DEVID_MinBufSz_Pos   6           /*!< TPI DEVID: MinBufSz Position */
#define TPI_DEVID_MinBufSz_Msk   (0x7UL << TPI_DEVID_MinBufSz_Pos)    /*!< TPI DEVID: MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos   5           /*!< TPI DEVID: AsynClkIn Position */
#define TPI_DEVID_AsynClkIn_Msk   (0x1UL << TPI_DEVID_AsynClkIn_Pos)    /*!< TPI DEVID: AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos 0           /*!< TPI DEVID: NrTraceInput Position */
#define TPI_DEVID_NrTraceInput_Msk (0xFUL /*<< TPI_DEVID_NrTraceInput_Pos*/) /*!< TPI DEVID: NrTraceInput Mask */

```

```

/* TPI DEVTYPE Register Definitions */

#define TPI_DEVTYPE_MajorType_Pos      4          /*!< TPI DEVTYPE: MajorType Position */
#define TPI_DEVTYPE_MajorType_Msk      (0xFUL << TPI_DEVTYPE_MajorType_Pos) /*!< TPI DEVTYPE: MajorType Mask */

#define TPI_DEVTYPE_SubType_Pos       0          /*!< TPI DEVTYPE: SubType Position */
#define TPI_DEVTYPE_SubType_Msk      (0xFUL /*<< TPI_DEVTYPE_SubType_Pos*/) /*!< TPI DEVTYPE: SubType Mask */

/*}@*/ /* end of group CMSIS_TPI */

```

```

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_MPUs Memory Protection Unit (MPU)
 \brief Type definitions for the Memory Protection Unit (MPU)

@{

*/

/** \brief Structure type to access the Memory Protection Unit (MPU).
 */

typedef struct
{
    __I uint32_t TYPE;           /*!< Offset: 0x000 (R/ ) MPU Type Register */
    __IO uint32_t CTRL;          /*!< Offset: 0x004 (R/W) MPU Control Register */
    __IO uint32_t RNR;           /*!< Offset: 0x008 (R/W) MPU Region RNRber Register */
    __IO uint32_t RBAR;          /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
}
```

```

    __IO uint32_t RASR;           /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register
*/
    __IO uint32_t RBAR_A1;        /*!< Offset: 0x014 (R/W) MPU Alias 1 Region Base Address Register
*/
    __IO uint32_t RASR_A1;        /*!< Offset: 0x018 (R/W) MPU Alias 1 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A2;        /*!< Offset: 0x01C (R/W) MPU Alias 2 Region Base Address Register
*/
    __IO uint32_t RASR_A2;        /*!< Offset: 0x020 (R/W) MPU Alias 2 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A3;        /*!< Offset: 0x024 (R/W) MPU Alias 3 Region Base Address Register
*/
    __IO uint32_t RASR_A3;        /*!< Offset: 0x028 (R/W) MPU Alias 3 Region Attribute and Size
Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)      /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)      /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)      /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

```

```

#define MPU_CTRL_PRIVDEFENA_Pos      2           /*!< MPU CTRL: PRIVDEFENA
Position */

#define MPU_CTRL_PRIVDEFENA_Msk     (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos      1           /*!< MPU CTRL: HFNMIENA
Position */

#define MPU_CTRL_HFNMIENA_Msk     (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos        0           /*!< MPU CTRL: ENABLE Position */

#define MPU_CTRL_ENABLE_Msk       (1UL /*<< MPU_CTRL_ENABLE_Pos*/)    /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */

#define MPU_RNR_REGION_Pos        0           /*!< MPU RNR: REGION Position */

#define MPU_RNR_REGION_Msk       (0xFFUL /*<< MPU_RNR_REGION_Pos*/)    /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos         5           /*!< MPU RBAR: ADDR Position */

#define MPU_RBAR_ADDR_Msk       (0x7FFFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos        4           /*!< MPU RBAR: VALID Position */

#define MPU_RBAR_VALID_Msk       (1UL << MPU_RBAR_VALID_Pos)    /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos        0           /*!< MPU RBAR: REGION Position */

#define MPU_RBAR_REGION_Msk       (0xFUL /*<< MPU_RBAR_REGION_Pos*/)    /*!< MPU
RBAR: REGION Mask */

```

```

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos      16          /*!< MPU RASR: MPU Region
Attribute field Position */

#define MPU_RASR_ATTRS_Msk      (0xFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28          /*!< MPU RASR: ATTRS.XN Position
*/
#define MPU_RASR_XN_Msk         (1UL << MPU_RASR_XN_Pos)        /*!< MPU RASR:
ATTRS.XN Mask */

#define MPU_RASR_AP_Pos         24          /*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk         (0x7UL << MPU_RASR_AP_Pos)       /*!< MPU RASR:
ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos        19          /*!< MPU RASR: ATTRS.TEX Position
*/
#define MPU_RASR_TEX_Msk        (0x7UL << MPU_RASR_TEX_Pos)       /*!< MPU RASR:
ATTRS.TEX Mask */

#define MPU_RASR_S_Pos           18          /*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk          (1UL << MPU_RASR_S_Pos)        /*!< MPU RASR: ATTRS.S
Mask */

#define MPU_RASR_C_Pos           17          /*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk          (1UL << MPU_RASR_C_Pos)        /*!< MPU RASR:
ATTRS.C Mask */

#define MPU_RASR_B_Pos           16          /*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk          (1UL << MPU_RASR_B_Pos)        /*!< MPU RASR:
ATTRS.B Mask */

```

```

#define MPU_RASR_SRD_Pos           8          /*!< MPU RASR: Sub-Region Disable
Position */

#define MPU_RASR_SRD_Msk          (0xFFUL << MPU_RASR_SRD_Pos)      /*!< MPU RASR:
Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos          1          /*!< MPU RASR: Region Size Field
Position */

#define MPU_RASR_SIZE_Msk          (0x1FUL << MPU_RASR_SIZE_Pos)      /*!< MPU RASR:
Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos         0          /*!< MPU RASR: Region enable bit
Position */

#define MPU_RASR_ENABLE_Msk         (1UL /*<< MPU_RASR_ENABLE_Pos*/)    /*!< MPU
RASR: Region enable bit Disable Mask */

/*}@} end of group CMSIS_MPUS
#endif

```

```

#if (__FPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
\defgroup CMSIS_FPU Floating Point Unit (FPU)
\brief Type definitions for the Floating Point Unit (FPU)

@{
*/

/** \brief Structure type to access the Floating Point Unit (FPU).
*/
typedef struct
{

```

```

    uint32_t RESERVED0[1];

    __IO uint32_t FPCCR;           /*!< Offset: 0x004 (R/W) Floating-Point Context Control Register */
    */

    __IO uint32_t FPCAR;          /*!< Offset: 0x008 (R/W) Floating-Point Context Address Register */
    */

    __IO uint32_t FPDSCR;         /*!< Offset: 0x00C (R/W) Floating-Point Default Status Control
Register      */

    __I uint32_t MVFR0;          /*!< Offset: 0x010 (R/ ) Media and FP Feature Register 0      */
    __I uint32_t MVFR1;          /*!< Offset: 0x014 (R/ ) Media and FP Feature Register 1      */

} FPU_Type;

/* Floating-Point Context Control Register */

#define FPU_FPCCR_AS PEN_Pos      31           /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk       (1UL << FPU_FPCCR_AS PEN_Pos)      /*!< FPCCR: ASPEN
bit Mask */

#define FPU_FPCCR_LSP EN_Pos      30           /*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSP EN_Msk       (1UL << FPU_FPCCR_LSP EN_Pos)      /*!< FPCCR: LSPEN
bit Mask */

#define FPU_FPCCR_MON RDY_Pos     8            /*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MON RDY_Msk     (1UL << FPU_FPCCR_MON RDY_Pos)      /*!< FPCCR:
MONRDY bit Mask */

#define FPU_FPCCR_BFR DY_Pos      6            /*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFR DY_Msk       (1UL << FPU_FPCCR_BFR DY_Pos)      /*!< FPCCR: BFRDY
bit Mask */

#define FPU_FPCCR_MMR DY_Pos      5            /*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMR DY_Msk       (1UL << FPU_FPCCR_MMR DY_Pos)      /*!< FPCCR:
MMRDY bit Mask */

```

```

#define FPU_FPCCR_HFRDY_Pos      4          /*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk      (1UL << FPU_FPCCR_HFRDY_Pos)    /*!< FPCCR:
HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos     3          /*!< FPCCR: processor mode bit
Position */
#define FPU_FPCCR_THREAD_Msk     (1UL << FPU_FPCCR_THREAD_Pos)    /*!< FPCCR:
processor mode active bit Mask */

#define FPU_FPCCR_USER_Pos       1          /*!< FPCCR: privilege level bit
Position */
#define FPU_FPCCR_USER_Msk       (1UL << FPU_FPCCR_USER_Pos)    /*!< FPCCR:
privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos     0          /*!< FPCCR: Lazy state preservation
active bit Position */
#define FPU_FPCCR_LSPACT_Msk     (1UL /*<< FPU_FPCCR_LSPACT_Pos*/) /*!< FPCCR: Lazy
state preservation active bit Mask */

/* Floating-Point Context Address Register */

#define FPU_FPCAR_ADDRESS_Pos    3          /*!< FPCAR: ADDRESS bit Position */
#define FPU_FPCAR_ADDRESS_Msk    (0x1FFFFFFFUL << FPU_FPCAR_ADDRESS_Pos) /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */

#define FPU_FPDSCR_AHP_Pos       26         /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk       (1UL << FPU_FPDSCR_AHP_Pos)    /*!< FPDSCR: AHP
bit Mask */

#define FPU_FPDSCR_DN_Pos        25         /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk        (1UL << FPU_FPDSCR_DN_Pos)    /*!< FPDSCR: DN bit
Mask */

```

```

#define FPU_FPDSCR_FZ_Pos      24           /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk      (1UL << FPU_FPDSCR_FZ_Pos)    /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos   22           /*!< FPDSCR: RMode bit Position */
#define FPU_FPDSCR_RMode_Msk   (3UL << FPU_FPDSCR_RMode_Pos)  /*!< FPDSCR:
RMode bit Mask */

/* Media and FP Feature Register 0 */

#define FPU_MVFRO_FP_rounding_modes_Pos 28          /*!< MVFRO: FP rounding
modes bits Position */
#define FPU_MVFRO_FP_rounding_modes_Msk (0xFUL << FPU_MVFRO_FP_rounding_modes_Pos)
/*!< MVFRO: FP rounding modes bits Mask */

#define FPU_MVFRO_Short_vectors_Pos 24           /*!< MVFRO: Short vectors bits
Position */
#define FPU_MVFRO_Short_vectors_Msk (0xFUL << FPU_MVFRO_Short_vectors_Pos) /*!<
MVFRO: Short vectors bits Mask */

#define FPU_MVFRO_Square_root_Pos 20           /*!< MVFRO: Square root bits
Position */
#define FPU_MVFRO_Square_root_Msk (0xFUL << FPU_MVFRO_Square_root_Pos) /*!<
MVFRO: Square root bits Mask */

#define FPU_MVFRO_Divide_Pos     16           /*!< MVFRO: Divide bits Position */
#define FPU_MVFRO_Divide_Msk    (0xFUL << FPU_MVFRO_Divide_Pos) /*!< MVFRO:
Divide bits Mask */

#define FPU_MVFRO_FP_excep_trapping_Pos 12        /*!< MVFRO: FP exception
trapping bits Position */

```

```

#define FPU_MVFR0_FP_excep_trapping_Msk  (0xFUL << FPU_MVFR0_FP_excep_trapping_Pos) /*!<
MVFR0: FP exception trapping bits Mask */

#define FPU_MVFR0_Double_precision_Pos    8                                /*!< MVFR0: Double-precision
bits Position */

#define FPU_MVFR0_Double_precision_Msk   (0xFUL << FPU_MVFR0_Double_precision_Pos) /*!<
MVFR0: Double-precision bits Mask */

#define FPU_MVFR0_Single_precision_Pos   4                                /*!< MVFR0: Single-precision
bits Position */

#define FPU_MVFR0_Single_precision_Msk  (0xFUL << FPU_MVFR0_Single_precision_Pos) /*!<
MVFR0: Single-precision bits Mask */

#define FPU_MVFR0_A SIMD_registers_Pos  0                                /*!< MVFR0: A SIMD registers
bits Position */

#define FPU_MVFR0_A SIMD_registers_Msk (0xFUL /*<< FPU_MVFR0_A SIMD_registers_Pos*/) /*!<
MVFR0: A SIMD registers bits Mask */

/* Media and FP Feature Register 1 */

#define FPU_MVFR1_FP_fused_MAC_Pos     28                               /*!< MVFR1: FP fused MAC
bits Position */

#define FPU_MVFR1_FP_fused_MAC_Msk   (0xFUL << FPU_MVFR1_FP_fused_MAC_Pos) /*!<
MVFR1: FP fused MAC bits Mask */

#define FPU_MVFR1_FP_HPFP_Pos        24                               /*!< MVFR1: FP HPFP bits
Position */

#define FPU_MVFR1_FP_HPFP_Msk       (0xFUL << FPU_MVFR1_FP_HPFP_Pos) /*!< MVFR1:
FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos     4                                /*!< MVFR1: D_NaN mode bits
Position */

#define FPU_MVFR1_D_NaN_mode_Msk   (0xFUL << FPU_MVFR1_D_NaN_mode_Pos) /*!<
MVFR1: D_NaN mode bits Mask */

```

```
#define FPU_MVFR1_FtZ_mode_Pos      0           /*!< MVFR1: FtZ mode bits
Position */

#define FPU_MVFR1_FtZ_mode_Msk      (0xFUL /*<< FPU_MVFR1_FtZ_mode_Pos*/) /*!<
MVFR1: FtZ mode bits Mask */

/*}@} end of group CMSIS_FPU */

#endif
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug    Core Debug Registers (CoreDebug)
\brief    Type definitions for the Core Debug Registers
@{
*/

/** \brief Structure type to access the Core Debug Register (CoreDebug).
*/
typedef struct
{
    __IO uint32_t DHCSR;          /*!< Offset: 0x000 (R/W) Debug Halting Control and Status Register */
    __O  uint32_t DCRSR;         /*!< Offset: 0x004 ( /W) Debug Core Register Selector Register */
    __IO uint32_t DCRDR;         /*!< Offset: 0x008 (R/W) Debug Core Register Data Register */
    __IO uint32_t DEMCR;         /*!< Offset: 0x00C (R/W) Debug Exception and Monitor Control
Register */
} CoreDebug_Type;

/* Debug Halting Control and Status Register */
```

```
#define CoreDebug_DHCSR_DBGKEY_Pos    16          /*!< CoreDebug DHCSR:  
DBGKEY Position */  
  
#define CoreDebug_DHCSR_DBGKEY_Msk    (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos)  /*!<  
CoreDebug DHCSR: DBGKEY Mask */  
  
  
#define CoreDebug_DHCSR_S_RESET_ST_Pos 25          /*!< CoreDebug DHCSR:  
S_RESET_ST Position */  
  
#define CoreDebug_DHCSR_S_RESET_ST_Msk (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RESET_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Pos 24          /*!< CoreDebug DHCSR:  
S_RETIRE_ST Position */  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Msk (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RETIRE_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_LOCKUP_Pos   19          /*!< CoreDebug DHCSR:  
S_LOCKUP Position */  
  
#define CoreDebug_DHCSR_S_LOCKUP_Msk  (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)  /*!<  
CoreDebug DHCSR: S_LOCKUP Mask */  
  
  
#define CoreDebug_DHCSR_S_SLEEP_Pos   18          /*!< CoreDebug DHCSR:  
S_SLEEP Position */  
  
#define CoreDebug_DHCSR_S_SLEEP_Msk  (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)  /*!<  
CoreDebug DHCSR: S_SLEEP Mask */  
  
  
#define CoreDebug_DHCSR_S_HALT_Pos    17          /*!< CoreDebug DHCSR:  
S_HALT Position */  
  
#define CoreDebug_DHCSR_S_HALT_Msk  (1UL << CoreDebug_DHCSR_S_HALT_Pos)  /*!<  
CoreDebug DHCSR: S_HALT Mask */  
  
  
#define CoreDebug_DHCSR_S_REGRDY_Pos  16          /*!< CoreDebug DHCSR:  
S_REGRDY Position */  
  
#define CoreDebug_DHCSR_S_REGRDY_Msk (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)  /*!<  
CoreDebug DHCSR: S_REGRDY Mask */
```

```

#define CoreDebug_DHCSR_C_SNAPSTALL_Pos  5          /*!< CoreDebug DHCSR: C_SNAPSTALL Position */

#define CoreDebug_DHCSR_C_SNAPSTALL_Msk  (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */


#define CoreDebug_DHCSR_C_MASKINTS_Pos   3          /*!< CoreDebug DHCSR: C_MASKINTS Position */

#define CoreDebug_DHCSR_C_MASKINTS_Msk  (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)
/*!< CoreDebug DHCSR: C_MASKINTS Mask */


#define CoreDebug_DHCSR_C_STEP_Pos      2          /*!< CoreDebug DHCSR: C_STEP Position */

#define CoreDebug_DHCSR_C_STEP_Msk     (1UL << CoreDebug_DHCSR_C_STEP_Pos)      /*!< CoreDebug DHCSR: C_STEP Mask */


#define CoreDebug_DHCSR_C_HALT_Pos     1          /*!< CoreDebug DHCSR: C_HALT Position */

#define CoreDebug_DHCSR_C_HALT_Msk    (1UL << CoreDebug_DHCSR_C_HALT_Pos)      /*!< CoreDebug DHCSR: C_HALT Mask */


#define CoreDebug_DHCSR_C_DEBUGEN_Pos   0          /*!< CoreDebug DHCSR: C_DEBUGEN Position */

#define CoreDebug_DHCSR_C_DEBUGEN_Msk  (1UL /*<< CoreDebug_DHCSR_C_DEBUGEN_Pos*/) /*!< CoreDebug DHCSR: C_DEBUGEN Mask */


/* Debug Core Register Selector Register */

#define CoreDebug_DCRSR_REGWnR_Pos    16         /*!< CoreDebug DCRSR: REGWnR Position */

#define CoreDebug_DCRSR_REGWnR_Msk   (1UL << CoreDebug_DCRSR_REGWnR_Pos)      /*!< CoreDebug DCRSR: REGWnR Mask */

```

```

#define CoreDebug_DCRSR_REGSEL_Pos      0           /*!< CoreDebug DCRSR: REGSEL Position */
#define CoreDebug_DCRSR_REGSEL_Msk     (0x1FUL /*<< CoreDebug_DCRSR_REGSEL_Pos*/) /*!< CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos    24          /*!< CoreDebug DEMCR: TRCENA Position */
#define CoreDebug_DEMCR_TRCENA_Msk   (1UL << CoreDebug_DEMCR_TRCENA_Pos) /*!< CoreDebug DEMCR: TRCENA Mask */

#define CoreDebug_DEMCR_MON_REQ_Pos   19          /*!< CoreDebug DEMCR: MON_REQ Position */
#define CoreDebug_DEMCR_MON_REQ_Msk  (1UL << CoreDebug_DEMCR_MON_REQ_Pos) /*!< CoreDebug DEMCR: MON_REQ Mask */

#define CoreDebug_DEMCR_MON_STEP_Pos  18          /*!< CoreDebug DEMCR: MON_STEP Position */
#define CoreDebug_DEMCR_MON_STEP_Msk (1UL << CoreDebug_DEMCR_MON_STEP_Pos) /*!< CoreDebug DEMCR: MON_STEP Mask */

#define CoreDebug_DEMCR_MON_PEND_Pos  17          /*!< CoreDebug DEMCR: MON_PEND Position */
#define CoreDebug_DEMCR_MON_PEND_Msk (1UL << CoreDebug_DEMCR_MON_PEND_Pos) /*!< CoreDebug DEMCR: MON_PEND Mask */

#define CoreDebug_DEMCR_MON_EN_Pos    16          /*!< CoreDebug DEMCR: MON_EN Position */
#define CoreDebug_DEMCR_MON_EN_Msk  (1UL << CoreDebug_DEMCR_MON_EN_Pos) /*!< CoreDebug DEMCR: MON_EN Mask */

#define CoreDebug_DEMCR_VC_HARDERR_Pos 10         /*!< CoreDebug DEMCR: VC_HARDERR Position */

```

```
#define CoreDebug_DEMCR_VC_HARDERR_Msk  (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos) /*!< CoreDebug DEMCR: VC_HARDERR Mask */

#define CoreDebug_DEMCR_VC_INTERR_Pos    9                         /*!< CoreDebug DEMCR: VC_INTERR Position */

#define CoreDebug_DEMCR_VC_INTERR_Msk   (1UL << CoreDebug_DEMCR_VC_INTERR_Pos) /*!< CoreDebug DEMCR: VC_INTERR Mask */

#define CoreDebug_DEMCR_VC_BUSERR_Pos   8                         /*!< CoreDebug DEMCR: VC_BUSERR Position */

#define CoreDebug_DEMCR_VC_BUSERR_Msk  (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos) /*!< CoreDebug DEMCR: VC_BUSERR Mask */

#define CoreDebug_DEMCR_VC_STATERR_Pos  7                         /*!< CoreDebug DEMCR: VC_STATERR Position */

#define CoreDebug_DEMCR_VC_STATERR_Msk (1UL << CoreDebug_DEMCR_VC_STATERR_Pos) /*!< CoreDebug DEMCR: VC_STATERR Mask */

#define CoreDebug_DEMCR_VC_CHKERR_Pos   6                         /*!< CoreDebug DEMCR: VC_CHKERR Position */

#define CoreDebug_DEMCR_VC_CHKERR_Msk  (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos) /*!< CoreDebug DEMCR: VC_CHKERR Mask */

#define CoreDebug_DEMCR_VC_NOCPERR_Pos  5                         /*!< CoreDebug DEMCR: VC_NOCPERR Position */

#define CoreDebug_DEMCR_VC_NOCPERR_Msk (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos) /*!< CoreDebug DEMCR: VC_NOCPERR Mask */

#define CoreDebug_DEMCR_VC_MMERR_Pos   4                         /*!< CoreDebug DEMCR: VC_MMERR Position */

#define CoreDebug_DEMCR_VC_MMERR_Msk  (1UL << CoreDebug_DEMCR_VC_MMERR_Pos) /*!< CoreDebug DEMCR: VC_MMERR Mask */
```

```

#define CoreDebug_DEMCR_VC_CORERESET_Pos 0          /*!< CoreDebug DEMCR:  

VC_CORERESET Position */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Msk (1UL /*<<  

CoreDebug_DEMCR_VC_CORERESET_Pos*/) /*!< CoreDebug DEMCR: VC_CORERESET Mask */  
  

/*@} end of group CMSIS_CoreDebug */  
  

  
  

/** \ingroup CMSIS_core_register  

\defgroup CMSIS_core_base Core Definitions  

\brief Definitions for base addresses, unions, and structures.  

@{  

*/  
  

/* Memory mapping of Cortex-M4 Hardware */  
  

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */  

#define ITM_BASE      (0xE0000000UL)           /*!< ITM Base Address */  

#define DWT_BASE      (0xE0001000UL)           /*!< DWT Base Address */  

#define TPI_BASE      (0xE0040000UL)           /*!< TPI Base Address */  

#define CoreDebug_BASE (0xE000EDFOUL)           /*!< Core Debug Base Address */  

#define SysTick_BASE  (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */  

#define NVIC_BASE     (SCS_BASE + 0x0100UL)       /*!< NVIC Base Address */  

#define SCB_BASE      (SCS_BASE + 0x0D00UL)       /*!< System Control Block Base Address */  
  

#define SCnSCB        ((SCnSCB_Type *) SCS_BASE) /*!< System control Register not in SCB */  

#define SCB           ((SCB_Type *) SCB_BASE)      /*!< SCB configuration struct */  

#define SysTick       ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */  

#define NVIC          ((NVIC_Type *) NVIC_BASE)      /*!< NVIC configuration struct */  

#define ITM           ((ITM_Type *) ITM_BASE)        /*!< ITM configuration struct */

```

```

#define DWT      ((DWT_Type *) DWT_BASE) /*!< DWT configuration struct */
#define TPI      ((TPI_Type *) TPI_BASE) /*!< TPI configuration struct */
#define CoreDebug ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration
struct */

#if (__MPU_PRESENT == 1)

#define MPU_BASE   (SCS_BASE + 0x0D90UL) /*!< Memory Protection Unit */
#define MPU       ((MPU_Type *) MPU_BASE) /*!< Memory Protection Unit */

#endif

#if (__FPU_PRESENT == 1)

#define FPU_BASE   (SCS_BASE + 0x0F30UL) /*!< Floating Point Unit */
#define FPU       ((FPU_Type *) FPU_BASE) /*!< Floating Point Unit */

#endif

/*@}*/

```

* Hardware Abstraction Layer

Core Function Interface contains:

- Core NVIC Functions
- Core SysTick Functions
- Core Debug Functions
- Core Register Access Functions

******/

/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference

*/

```

/* ##### NVIC functions #####
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the NVIC.

@{
*/

```

/** \brief Set Priority Grouping

The function sets the priority grouping field using the required unlock sequence.

The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.

Only values from 0..7 are used.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

\param [in] PriorityGroup Priority grouping field.

```

*/
__STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
    uint32_t reg_value;
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL);      /* only values 0..7 are used
*/
```

```

    reg_value = SCB->AIRCR;                      /* read old register configuration */
    reg_value &= ~((uint32_t)(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk)); /* clear
bits to change */
```

```

reg_value = (reg_value           |
((uint32_t)0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
(PriorityGroupTmp << 8)           );      /* Insert write key and priority group */
SCB->AIRCR = reg_value;
}

```

/** \brief Get Priority Grouping

The function reads the priority grouping field from the NVIC Interrupt Controller.

```

\return      Priority grouping field (SCB->AIRCR [10:8] PRIGROUP field).
*/
__STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)
{
    return ((uint32_t)((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos));
}

```

/** \brief Enable External Interrupt

The function enables a device-specific interrupt in the NVIC interrupt controller.

```

\param [in]    IRQn  External interrupt number. Value cannot be negative.
*/
__STATIC_INLINE void NVIC_EnableIRQ(int32_t IRQn)
{
    NVIC->ISER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &
0x1FUL));
}

```

```
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ICER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
0x1FUL));
```

```
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit
for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
```

```
{
```

```
    return((uint32_t)((NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0UL);  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

```
\param [in] IRQn Interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Get Active Interrupt
```

The function reads the active register in NVIC and returns the active bit.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not active.

\return 1 Interrupt status is active.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
```

```
{
```

```
    return((uint32_t)((NVIC->IABR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL)) != 0UL) ? 1UL : 0UL);
```

```
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

```
{
```

```
    if((int32_t)IRQn < 0) {
```

```

    SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &
(uint32_t)0xFFUL);

}

else {

    NVIC->IP[((uint32_t)(int32_t)IRQn)]      = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &
(uint32_t)0xFFUL);

}

}

```

/ \brief Get Interrupt Priority**

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

```

\param [in] IRQn Interrupt number.

\return      Interrupt Priority. Value is aligned automatically to the implemented
priority bits of the microcontroller.

*/
__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{

if((int32_t)IRQn < 0) {

    return(((uint32_t)SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL) >> (8 - __NVIC_PRIO_BITS));

}

else {

    return(((uint32_t)NVIC->IP[((uint32_t)(int32_t)IRQn)]      >> (8 - __NVIC_PRIO_BITS)));
}

```

```
}
```

```
/** \brief Encode Priority
```

The function encodes the priority for an interrupt with the given priority group, preemptive priority value, and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

```
\param [in] PriorityGroup Used priority group.  
\param [in] PreemptPriority Preemptive priority value (starting from 0).  
\param [in] SubPriority Subpriority value (starting from 0).  
\return Encoded priority. Value can be used in the function \ref NVIC_SetPriority().  
*/  
__STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority,  
uint32_t SubPriority)  
{  
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */  
    uint32_t PreemptPriorityBits;  
    uint32_t SubPriorityBits;  
  

```

```

        ((SubPriority & (uint32_t)((1UL << (SubPriorityBits )) - 1UL)))
    );
}

```

*/** \brief Decode Priority*

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

`\param [in] Priority` Priority value, which can be retrieved with the function `\ref NVIC_GetPriority()`.

`\param [in] PriorityGroup` Used priority group.

`\param [out] pPreemptPriority` Preemptive priority value (starting from 0).

`\param [out] pSubPriority` Subpriority value (starting from 0).

`*/`

`__STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)`

`{`

`uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */`

`uint32_t PreemptPriorityBits;`

`uint32_t SubPriorityBits;`

`PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
 (uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);`

`SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
 (uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));`

```
*pPreemptPriority = (Priority >> SubPriorityBits) & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL);
*pSubPriority   = (Priority           ) & (uint32_t)((1UL << (SubPriorityBits )) - 1UL);
}
```

```
/** \brief System Reset
```

The function initiates a system reset request to reset the MCU.

```
*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                /* Ensure all outstanding memory accesses included
                                              buffered write are completed before reset */

    SCB->AIRCR = (uint32_t)((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                           (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
                           SCB_AIRCR_SYSRESETREQ_Msk);      /* Keep priority group unchanged */

    __DSB();                                /* Ensure completion of memory access */

    while(1) { __NOP(); }                    /* wait until reset */
}
```

```
/*@} end of CMSIS_Core_NVICFunctions */
```

```
/* ##### SysTick function
#####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.
```

```

@{

*/



#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0

/** \brief System Tick Configuration

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.
Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.
\return 1 Function failed.

\note When the variable <b>__Vendor_SysTickConfig</b> is set to 1, then the
function <b>SysTick_Config</b> is not included. In this case, the file <b><i>device</i>.h</b>
must contain a vendor-specific implementation of this function.

*/
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) { return (1UL); } /* Reload value impossible */

    SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
    SysTick->VAL = 0UL; /* Load the SysTick Counter Value */
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |

```

```
    SysTick_CTRL_TICKINT_Msk |  
    SysTick_CTRL_ENABLE_Msk;           /* Enable SysTick IRQ and SysTick Timer */  
return (0UL);                      /* Function successful */  
}
```

```
#endif
```

```
/*}@} end of CMSIS_Core_SysTickFunctions */
```

```
/* ##### Debug In/Output function  
##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface  
 \defgroup CMSIS_core_DebugFunctions ITM Functions  
 \brief Functions that access the ITM debug interface.  
 @{  
 */
```

```
extern volatile int32_t ITM_RxBuffer;          /*!< External variable to receive characters.  
 */
```

```
#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is  
ready for next character. */
```

```
/** \brief ITM Send Character
```

The function transmits a character via the ITM channel 0, and
\li Just returns when no debugger is connected that has booked the output.

\li Is blocking when a debugger is connected, but the previous character sent has not been transmitted.

\param [in] ch Character to transmit.

\returns Character to transmit.

*/

```
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)
{
    if (((ITM->TCR & ITM_TCR_ITMENA_Msk) != 0UL) && /* ITM enabled */
        ((ITM->TER & 1UL) != 0UL) ) /* ITM Port #0 enabled */
    {
        while (ITM->PORT[0].u32 == 0UL) { __NOP(); }

        ITM->PORT[0].u8 = (uint8_t)ch;
    }
    return (ch);
}
```

/** \brief ITM Receive Character

The function inputs a character via the external variable \ref ITM_RxBuffer.

\return Received character.

\return -1 No character pending.

*/

```
__STATIC_INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1; /* no character available */
```

```

if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
    ch = ITM_RxBuffer;
    ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
}

return (ch);
}

```

/** \brief ITM Check Character

The function checks whether a character is pending for reading in the variable \ref ITM_RxBuffer.

```

\return 0 No character available.

\return 1 Character available.

*/
__STATIC_INLINE int32_t ITM_CheckChar (void) {

if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {
    return (0); /* no character available */
} else {
    return (1); /* character available */
}

/*@} end of CMSIS_core_DebugFunctions */

```

```
#ifdef __cplusplus
}

#endif /* __CORE_CM4_H_DEPENDANT */

#endif /* __CMSIS_GENERIC */

/********************* @file core_cmFunc.h
 * @brief CMSIS Cortex-M Core Function Access Header File
 * @version V4.10
 * @date 18. March 2015
 *
 * @note
 *
 ****/
/* Copyright (c) 2009 - 2015 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMFUNC_H
#define __CORE_CMFUNC_H

/* ##### Core Function Access #####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_RegAccFunctions CMSIS Core Register Access Functions
 @{
 */

#if defined (__CC_ARM) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */
```

```
#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif
```

```
/* intrinsic void __enable_irq(); */
/* intrinsic void __disable_irq(); */
```

```
/** \brief Get Control Register
```

This function returns the content of the Control Register.

```
\return Control Register value
*/
__STATIC_INLINE uint32_t __get_CONTROL(void)
{
    register uint32_t __regControl      __ASM("control");
    return(__regControl);
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in] control Control Register value to set
*/
__STATIC_INLINE void __set_CONTROL(uint32_t control)
{
    register uint32_t __regControl      __ASM("control");
```

```
__regControl = control;  
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return IPSR Register value  
*/  
STATIC_INLINE uint32_t __get_IPSR(void)  
{  
    register uint32_t __regIPSR      __ASM("ipsr");  
    return(__regIPSR);  
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return APSR Register value  
*/  
STATIC_INLINE uint32_t __get_APSR(void)  
{  
    register uint32_t __regAPSR      __ASM("apsr");  
    return(__regAPSR);  
}
```

```
/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
\return xPSR Register value
*/
__STATIC_INLINE uint32_t __get_xPSR(void)
{
    register uint32_t __regXPSR      __ASM("xpsr");
    return(__regXPSR);
}
```

```
/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
\return PSP Register value
*/
__STATIC_INLINE uint32_t __get_PSP(void)
{
    register uint32_t __regProcessStackPointer __ASM("psp");
    return(__regProcessStackPointer);
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in] topOfProcStack Process Stack Pointer value to set  
*/  
STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)  
{  
    register uint32_t __regProcessStackPointer __ASM("psp");  
    __regProcessStackPointer = topOfProcStack;  
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value  
*/  
STATIC_INLINE uint32_t __get_MSP(void)  
{  
    register uint32_t __regMainStackPointer __ASM("msp");  
    return(__regMainStackPointer);  
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
```

```
*/  
  
__STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)  
{  
    register uint32_t __regMainStackPointer __ASM("msp");  
    __regMainStackPointer = topOfMainStack;  
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value  
*/  
  
__STATIC_INLINE uint32_t __get_PRIMASK(void)  
{  
    register uint32_t __regPriMask __ASM("primask");  
    return(__regPriMask);  
}
```

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in] priMask Priority Mask  
*/  
  
__STATIC_INLINE void __set_PRIMASK(uint32_t priMask)  
{
```

```
register uint32_t __regPriMask      __ASM("primask");  
__regPriMask = (priMask);  
}
```

```
#if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
#define __enable_fault_irq      __enable_fiq
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
#define __disable_fault_irq     __disable_fiq
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

\return Base Priority register value

```
*/
```

```
__STATIC_INLINE uint32_t __get_BASEPRI(void)
{
    register uint32_t __regBasePri      __ASM("basepri");
    return(__regBasePri);
}
```

/** \brief Set Base Priority

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set
*/
__STATIC_INLINE void __set_BASEPRI(uint32_t basePri)
{
    register uint32_t __regBasePri      __ASM("basepri");
    __regBasePri = (basePri & 0xff);
}
```

/** \brief Set Base Priority with condition

This function assigns the given value to the Base Priority register only if BASEPRI masking is disabled, or the new value increases the BASEPRI priority level.

```
\param [in] basePri Base Priority value to set
*/
__STATIC_INLINE void __set_BASEPRI_MAX(uint32_t basePri)
{
```

```
register uint32_t __regBasePriMax    __ASM("basepri_max");
__regBasePriMax = (basePri & 0xff);
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return      Fault Mask register value
*/
__STATIC_INLINE uint32_t __get_FAULTMASK(void)
{
    register uint32_t __regFaultMask    __ASM("faultmask");
    return(__regFaultMask);
}
```

```
/** \brief Set Fault Mask
```

This function assigns the given value to the Fault Mask register.

```
\param [in]  faultMask  Fault Mask value to set
*/
__STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    register uint32_t __regFaultMask    __ASM("faultmask");
    __regFaultMask = (faultMask & (uint32_t)1);
}
```

```
#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */
```

```
#if    (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)
```

```
/** \brief Get FPSCR
```

This function returns the current value of the Floating Point Status/Control register.

```
\return      Floating Point Status/Control register value
*/
__STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpscr      __ASM("fpscr");
    return(__regfpscr);
#else
    return(0);
#endif
}
```

```
/** \brief Set FPSCR
```

This function assigns the given value to the Floating Point Status/Control register.

```
\param [in]  fpscr Floating Point Status/Control value to set
*/
```

```

__STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpSCR      __ASM("fpSCR");
    __regfpSCR = (fpSCR);
#endif
}

#endif /* (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07) */

#ifndef __GNUC__
/*----- GNU Compiler -----*/
/* GNU gcc specific functions */

/** \brief Enable IRQ Interrupts
 */

```

This function enables IRQ interrupts by clearing the I-bit in the CPSR.

Can only be executed in Privileged modes.

```

__attribute__(( always_inline )) __STATIC_INLINE void __enable_irq(void)
{
    __ASM volatile ("cpsie i" : : : "memory");
}

```

```

/** \brief Disable IRQ Interrupts
 */

```

This function disables IRQ interrupts by setting the I-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_irq(void)  
{  
    __ASM volatile ("cpsid i" : : : "memory");  
}
```

```
/** \brief Get Control Register
```

This function returns the content of the Control Register.

```
\return      Control Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_CONTROL(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, control" : "=r" (result));  
    return(result);  
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in]  control Control Register value to set  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_CONTROL(uint32_t control)
```

```
{  
    __ASM volatile ("MSR control, %0" :: "r" (control) : "memory");  
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return      IPSR Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_IPSR(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, ipsr" : "=r" (result) );  
    return(result);  
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return      APSR Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_APSR(void)  
{  
    uint32_t result;
```

```
    __ASM volatile ("MRS %0, apsr" : "=r" (result) );
    return(result);
}
```

```
/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
    \return      xPSR Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_xPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, xpsr" : "=r" (result) );
    return(result);
}
```

```
/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
    \return      PSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_PSP(void)
{
```

```
register uint32_t result;

__ASM volatile ("MRS %0, psp\n" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in] topOfProcStack Process Stack Pointer value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)
{
    __ASM volatile ("MSR psp, %0\n" :: "r" (topOfProcStack) : "sp");
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t result;
```

```
__ASM volatile ("MRS %0, msp\n" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)
{
    __ASM volatile ("MSR msp, %0\n" :: "r" (topOfMainStack) : "sp");
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_PRIMASK(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, primask" : "=r" (result) );
    return(result);
```

```
}
```

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in] priMask Priority Mask
```

```
*/
```

```
__attribute__(( always_inline )) __STATIC_INLINE void __set_PRIMASK(uint32_t priMask)
```

```
{
```

```
__ASM volatile ("MSR primask, %0" :: "r" (priMask) : "memory");
```

```
}
```

```
#if __CORTEX_M >= 0x03
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
__attribute__(( always_inline )) __STATIC_INLINE void __enable_fault_irq(void)
```

```
{
```

```
__ASM volatile ("cpsie f" :: : "memory");
```

```
}
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_fault_irq(void)  
{  
    __ASM volatile ("cpsid f" :: : "memory");  
}
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

```
\return      Base Priority register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_BASEPRI(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, basepri" : "=r" (result));  
    return(result);  
}
```

```
/** \brief Set Base Priority
```

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set  
*/  
attribute__( always_inline ) STATIC_INLINE void __set_BASEPRI(uint32_t value)  
{  
    __ASM volatile ("MSR basepri, %0" :: "r" (value) : "memory");  
}
```

```
/** \brief Set Base Priority with condition
```

This function assigns the given value to the Base Priority register only if BASEPRI masking is disabled, or the new value increases the BASEPRI priority level.

```
\param [in] basePri Base Priority value to set  
*/  
attribute__( always_inline ) STATIC_INLINE void __set_BASEPRI_MAX(uint32_t value)  
{  
    __ASM volatile ("MSR basepri_max, %0" :: "r" (value) : "memory");  
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return Fault Mask register value  
*/  
attribute__( always_inline ) STATIC_INLINE uint32_t __get_FAULTMASK(void)  
{
```

```
uint32_t result;

__ASM volatile ("MRS %0, faultmask" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Fault Mask
```

This function assigns the given value to the Fault Mask register.

```
\param [in] faultMask Fault Mask value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    __ASM volatile ("MSR faultmask, %0" : : "r" (faultMask) : "memory");
}

#endif /* (__CORTEX_M >= 0x03) */
```

```
#if    (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)
```

```
/** \brief Get FPSCR
```

This function returns the current value of the Floating Point Status/Control register.

```
\return      Floating Point Status/Control register value
*/
```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    uint32_t result;

    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMRS %0, fpSCR" : "=r" (result));
    __ASM volatile ("");
    return(result);
#else
    return(0);
#endif
}

```

/** \brief Set FPSCR

This function assigns the given value to the Floating Point Status/Control register.

```

\param [in] fpSCR Floating Point Status/Control value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMSR fpSCR, %0" : : "r" (fpSCR) : "vfpcc");
    __ASM volatile ("");

```

```
#endif
}

#endif /* (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07) */

#ifndef __CMSIS_H
#define __CMSIS_H

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
 */
#endif /* __CMSIS_H */
```



```
#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>
```

```
#endif

/*@} end of CMSIS_Core_RegAccFunctions */

#endif /* __CORE_CMFUNC_H */

/*********************//**/

* @file core_cm4_simd.h
* @brief CMSIS Cortex-M4 SIMD Header File
* @version V3.30
* @date 17. February 2014
*
* @note
*
*****/
```

/* Copyright (c) 2009 - 2014 ARM LIMITED

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif

#ifndef __CORE_CM4 SIMD_H
#define __CORE_CM4 SIMD_H

#endif /* cplusplus

extern "C" {

#endif

/*********************  

*      Hardware Abstraction Layer  

********************/  


```

```
/* ##### Compiler specific Intrinsics ##### */
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
Access to dedicated SIMD instructions

@{

*/
#ifndef __CMSIS_SIMD_H
#define __CMSIS_SIMD_H

/*if defined (__CC_ARM) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#define __SADD8          __sadd8
#define __QADD8          __qadd8
#define __SHADD8         __shadd8
#define __UADD8          __uadd8
#define __UQADD8         __uqadd8
#define __UHADD8         __uhadd8
#define __SSUB8          __ssub8
#define __QSUB8          __qsub8
#define __SHSUB8         __shsub8
#define __USUB8          __usub8
#define __UQSUB8         __uqsub8
#define __UHSUB8         __uhsub8
#define __SADD16         __sadd16
#define __QADD16         __qadd16
#define __SHADD16        __shadd16
#define __UADD16         __uadd16
#define __UQADD16        __uqadd16
#define __UHADD16        __uhadd16
#define __SSUB16         __ssub16
#define __QSUB16         __qsub16

#endif /* __CMSIS_SIMD_H */
```

```
#define __SHSUB16          __shsub16
#define __USUB16           __usub16
#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX              __sasx
#define __QASX              __qasx
#define __SHASX             __shasx
#define __UASX              __uasx
#define __UQASX             __uqasx
#define __UHASX             __uhasx
#define __SSAX               __ssax
#define __QSAX               __qsax
#define __SHSAX              __shsax
#define __USAX               __usax
#define __UQSAX              __uqsax
#define __UHSAX              __uhsax
#define __USAD8              __usad8
#define __USADA8             __usada8
#define __SSAT16             __ssat16
#define __USAT16             __usat16
#define __UXTB16             __uxtb16
#define __UXTAB16            __uxtab16
#define __SXTB16             __sxtb16
#define __SXTAB16            __sxtab16
#define __SMUAD              __smuad
#define __SMUADX             __smuadx
#define __SMLAD              __smlad
#define __SMLADX             __smladx
#define __SMLALD             __smlald
```

```

#define __SMLALDX      __smlaldx
#define __SMUSD         __smusd
#define __SMUSDX        __smusdx
#define __SMLSD          __smlsd
#define __SMLSDX         __smlsdx
#define __SMLS LD        __smlsld
#define __SMLS LDX       __smlsldx
#define __SEL            __sel
#define __QADD           __qadd
#define __QSUB           __qsub

#define __PKHBT(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0x0000FFFFUL) | \
                                (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0xFFFF0000UL) | \
                                (((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL) )

#define __SMMLA(ARG1,ARG2,ARG3)    ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                ((int64_t)(ARG3) << 32) ) >> 32))

# elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB16(uint32_t op1, uint32_t op2)
{
```

```

    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAX(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSAX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSAX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAD8(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```

}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __USADA8(uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#define __SSAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

#define __USAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTB16(uint32_t op1)
{
    uint32_t result;

```

```
__ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));  
return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UXTAB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTB16(uint32_t op1)  
{  
    uint32_t result;  
  
    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUAD (uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUADX (uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLAD (uint32_t op1, uint32_t op2,  
uint32_t op3)  
{  
    uint32_t result;  
  
    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLADX (uint32_t op1, uint32_t op2,  
uint32_t op3)  
{  
    uint32_t result;
```

```

__ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
}

```

```

} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2), "0"
(llr.w32[0]), "1" (llr.w32[1]));
#else // Big endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2), "0"
(llr.w32[1]), "1" (llr.w32[0]));
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMUSD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMUSDX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD (uint32_t op1, uint32_t op2,
uint32_t op3)

{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));

    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX (uint32_t op1, uint32_t op2,
uint32_t op3)

{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));

    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSDFD (uint32_t op1, uint32_t op2,
uint64_t acc)

{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#endif __ARMEB__ // Little endian

```

```

__ASM volatile ("smllsd %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else      // Big endian

__ASM volatile ("smllsd %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLSLDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{

    uint32_t w32[2];

    uint64_t w64;

} llr;

llr.w64 = acc;

#endifdef __ARMEB__ // Little endian

__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else      // Big endian

__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SEL( uint32_t op1, uint32_t op2 )
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QADD(uint32_t op1, uint32_t op2)
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QSUB(uint32_t op1, uint32_t op2)
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
#define __PKHBT(ARG1,ARG2,ARG3) \
```

```
{ \
```

```
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
```

```
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
```

```
    __RES; \
```

```
})
```

```
#define __PKHTB(ARG1,ARG2,ARG3) \
{ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2) ); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "I" (ARG3) ); \
    __RES; \
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMMLA (int32_t op1, int32_t op2, int32_t
op3)
{
    int32_t result;

    __ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
```

```
#include <cmsis_ccs.h>

#if defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/* not yet supported */

#if defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD_intrinsics */

#ifndef __cplusplus
}

#endif /* __CORE_CM4 SIMD_H */
//********************************************************************//**
* @file core_cm7.h
* @brief CMSIS Cortex-M7 Core Peripheral Access Layer Header File
* @version V4.10
* @date 18. March 2015
*
* @note
```

*

******/

/* Copyright (c) 2009 - 2015 ARM LIMITED

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif
```

```
#ifndef __CORE_CM7_H_GENERIC

#define __CORE_CM7_H_GENERIC
```

```
#ifdef __cplusplus

extern "C" {

#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions

CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '...'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.
*/

```
*****  
*      CMSIS definitions  
*****/*
```

```

/** \ingroup Cortex_M7

@{

*/


/* CMSIS CM7 definitions */

#define __CM7_CMSIS_VERSION_MAIN (0x04)           /*!< [31:16] CMSIS HAL main version */
#define __CM7_CMSIS_VERSION_SUB (0x00)             /*!< [15:0] CMSIS HAL sub version */
#define __CM7_CMSIS_VERSION ((__CM7_CMSIS_VERSION_MAIN << 16) | \
                           __CM7_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */

#define __CORTEX_M      (0x07)          /*!< Cortex-M Core */

#if defined (__CC_ARM)

#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)

#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)

#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

```

```

#define __ASM      __asm          /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#define __ASM      __asm          /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline         /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

#define __packed

#define __ASM      __asm          /*!< asm keyword for COSMIC Compiler */
#define __INLINE   inline         /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */

#define __STATIC_INLINE static inline

#endif

```

/** __FPU_USED indicates whether an FPU is used or not.

For this, __FPU_PRESENT has to be checked prior to making use of FPU specific registers and functions.

```

*/
#if defined ( __CC_ARM )
#if defined __TARGET_FPU_VFP
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"

```

```

#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif

#elif defined ( __GNUC__ )
#if defined ( __VFP_FP__ ) && !defined( __SOFTFP__ )
#if ( __FPU_PRESENT == 1 )
#define __FPU_USED 1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif

#elif defined ( __ICCARM__ )
#if defined __ARMVFP__
#if ( __FPU_PRESENT == 1 )
#define __FPU_USED 1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED 0
#endif
#else

```

```
#define __FPU_USED    0
#endif

#ifndef __TMS470__
#define __FPU_USED    0
#endif

#define __FPU_USED    0
#endif
```

```

#if ( __CSMC__ & 0x400)           // FPU present for parser

#if ( __FPU_PRESENT == 1)

#define __FPU_USED    1

#else

#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

#define __FPU_USED    0

#endif

#endif

#define __FPU_USED    0

#endif

#endif

#include <stdint.h>          /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access      */
#include <core_cmFunc.h>         /* Core Function Access      */
#include <core_cmSimd.h>         /* Compiler specific SIMD Intrinsics */

#endif /* __cplusplus

}

#endif /* __CORE_CM7_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM7_H_DEPENDANT

#define __CORE_CM7_H_DEPENDANT

#endif /* __cplusplus

```

```
extern "C" {

#endif

/* check device defines and use defaults */

#if defined __CHECK_DEVICE_DEFINES

#ifndef __CM7_REV
#define __CM7_REV      0x0000
#warning "__CM7_REV not defined in device header file; using default!"

#endif

#ifndef __FPU_PRESENT
#define __FPU_PRESENT    0
#warning "__FPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT    0
#warning "__MPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __ICACHE_PRESENT
#define __ICACHE_PRESENT 0
#warning "__ICACHE_PRESENT not defined in device header file; using default!"

#endif

#ifndef __DCACHE_PRESENT
#define __DCACHE_PRESENT 0
#warning "__DCACHE_PRESENT not defined in device header file; using default!"

#endif
```

```

#ifndef __DTCM_PRESENT
#define __DTCM_PRESENT      0
#warning "__DTCM_PRESENT      not defined in device header file; using default!"
#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      3
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig  0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif

#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
 \defgroup CMSIS_glob_defs CMSIS Global Defines

 <strong>IO Type Qualifiers</strong> are used
 \li to specify the access to peripheral variables.
 \li for automatic generation of peripheral register debug information.
 */

#ifndef __cplusplus
#define __I volatile      /*!< Defines 'read only' permissions      */
#else
#define __I volatile const /*!< Defines 'read only' permissions      */

```

```
#endif

#define __O volatile /*!< Defines 'write only' permissions */
#define __IO volatile /*!< Defines 'read / write' permissions */

/*@} end of group Cortex_M7 */
```

```
*****
```

```
* Register Abstraction
```

```
Core Register contain:
```

- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core Debug Register
- Core MPU Register
- Core FPU Register

```
*****/
```

```
/** \defgroup CMSIS_core_register Defines and Type Definitions
 \brief Type definitions and defines for Cortex-M processor based devices.
 */


```

```
/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.

 @{
 */


```

```

/** \brief Union type to access the Application Program Status Register (APSR).
*/
typedef union
{
    struct
    {
        uint32_t _reserved0:16;           /*!< bit: 0..15 Reserved          */
        uint32_t GE:4;                  /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:7;           /*!< bit: 20..26 Reserved          */
        uint32_t Q:1;                  /*!< bit: 27 Saturation condition flag */
        uint32_t V:1;                  /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;                  /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;                  /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;                  /*!< bit: 31 Negative condition code flag */
    } b;                         /*!< Structure used for bit access      */
    uint32_t w;                   /*!< Type used for word access         */
} APSR_Type;

/* APSR Register Definitions */

#define APSR_N_Pos            31           /*!< APSR: N Position */
#define APSR_N_Msk             (1UL << APSR_N_Pos)      /*!< APSR: N Mask */

#define APSR_Z_Pos            30           /*!< APSR: Z Position */
#define APSR_Z_Msk             (1UL << APSR_Z_Pos)      /*!< APSR: Z Mask */

#define APSR_C_Pos            29           /*!< APSR: C Position */
#define APSR_C_Msk             (1UL << APSR_C_Pos)      /*!< APSR: C Mask */

#define APSR_V_Pos            28           /*!< APSR: V Position */

```

```

#define APSR_V_Msk          (1UL << APSR_V_Pos)           /*!< APSR: V Mask */

#define APSR_Q_Pos           27                         /*!< APSR: Q Position */
#define APSR_Q_Msk          (1UL << APSR_Q_Pos)           /*!< APSR: Q Mask */

#define APSR_GE_Pos          16                         /*!< APSR: GE Position */
#define APSR_GE_Msk         (0xFUL << APSR_GE_Pos)        /*!< APSR: GE Mask */

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

*/

```

typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;               /*!< Type used for word access */
} IPSR_Type;

```

/* IPSR Register Definitions */

```

#define IPSR_ISR_Pos          0                         /*!< IPSR: ISR Position */
#define IPSR_ISR_Msk         (0x1FFUL /*<< IPSR_ISR_Pos*/)      /*!< IPSR: ISR Mask */

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).

*/

```

typedef union

```

```

{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:7;    /*!< bit: 9..15 Reserved */
        uint32_t GE:4;           /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:4;    /*!< bit: 20..23 Reserved */
        uint32_t T:1;             /*!< bit: 24 Thumb bit (read 0) */
        uint32_t IT:2;            /*!< bit: 25..26 saved IT state (read 0) */
        uint32_t Q:1;              /*!< bit: 27 Saturation condition flag */
        uint32_t V:1;              /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;              /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;              /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;              /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

```

/ xPSR Register Definitions */*

#define xPSR_N_Pos	31	/*!< xPSR: N Position */
#define xPSR_N_Msk	(1UL << xPSR_N_Pos)	/*!< xPSR: N Mask */
#define xPSR_Z_Pos	30	/*!< xPSR: Z Position */
#define xPSR_Z_Msk	(1UL << xPSR_Z_Pos)	/*!< xPSR: Z Mask */
#define xPSR_C_Pos	29	/*!< xPSR: C Position */
#define xPSR_C_Msk	(1UL << xPSR_C_Pos)	/*!< xPSR: C Mask */
#define xPSR_V_Pos	28	/*!< xPSR: V Position */

#define xPSR_V_Msk	(1UL << xPSR_V_Pos)	/*!< xPSR: V Mask */
#define xPSR_Q_Pos	27	/*!< xPSR: Q Position */
#define xPSR_Q_Msk	(1UL << xPSR_Q_Pos)	/*!< xPSR: Q Mask */
#define xPSR_IT_Pos	25	/*!< xPSR: IT Position */
#define xPSR_IT_Msk	(3UL << xPSR_IT_Pos)	/*!< xPSR: IT Mask */
#define xPSR_T_Pos	24	/*!< xPSR: T Position */
#define xPSR_T_Msk	(1UL << xPSR_T_Pos)	/*!< xPSR: T Mask */
#define xPSR_GE_Pos	16	/*!< xPSR: GE Position */
#define xPSR_GE_Msk	(0xFUL << xPSR_GE_Pos)	/*!< xPSR: GE Mask */
#define xPSR_ISR_Pos	0	/*!< xPSR: ISR Position */
#define xPSR_ISR_Msk	(0x1FFUL /*<< xPSR_ISR_Pos*/)	/*!< xPSR: ISR Mask */

```
/** \brief Union type to access the Control Registers (CONTROL).
 */
typedef union
{
    struct
    {
        uint32_t nPRIV:1;          /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                         /*!< Structure used for bit access */
}
```

```

    uint32_t w;           /*!< Type used for word access */
} CONTROL_Type;

/* CONTROL Register Definitions */

#define CONTROL_FPCA_Pos      2           /*!< CONTROL: FPCA Position */
#define CONTROL_FPCA_Msk      (1UL << CONTROL_FPCA_Pos)      /*!< CONTROL: FPCA
Mask */

#define CONTROL_SPSEL_Pos     1           /*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk     (1UL << CONTROL_SPSEL_Pos)      /*!< CONTROL: SPSEL
Mask */

#define CONTROL_nPRIV_Pos     0           /*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk     (1UL /*<< CONTROL_nPRIV_Pos*/)      /*!< CONTROL:
nPRIV Mask */

/*@} end of group CMSIS_CORE */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
@{
*/

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[8];      /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */

```

```

    uint32_t RESERVED0[24];

    __IO uint32_t ICER[8];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
    uint32_t RSERVED1[24];

    __IO uint32_t ISPR[8];          /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */
    uint32_t RESERVED2[24];

    __IO uint32_t ICPR[8];          /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */
    uint32_t RESERVED3[24];

    __IO uint32_t IABR[8];          /*!< Offset: 0x200 (R/W) Interrupt Active bit Register */
    uint32_t RESERVED4[56];

    __IO uint8_t IP[240];           /*!< Offset: 0x300 (R/W) Interrupt Priority Register (8Bit wide) */

    uint32_t RESERVED5[644];

    __O uint32_t STIR;             /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register */
} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */

#define NVIC_STIR_INTID_Pos        0                      /*!< STIR: INTLINESNUM Position */
#define NVIC_STIR_INTID_Msk        (0x1FFUL /*<< NVIC_STIR_INTID_Pos*/)      /*!< STIR:
INTLINESNUM Mask */

```

```
/*@} end of group CMSIS_NVIC */
```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers
@{
*/

```

```
/** \brief Structure type to access the System Control Block (SCB).
```

```

*/
typedef struct
{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
/*/
    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
    __IO uint32_t AIRCR;         /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register   */
    __IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register          */
    __IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register    */
    __IO uint8_t SHPR[12];        /*!< Offset: 0x018 (R/W) System Handlers Priority Registers (4-7, 8-11,
12-15) */
    __IO uint32_t SHCSR;          /*!< Offset: 0x024 (R/W) System Handler Control and State Register
*/
    __IO uint32_t CFSR;           /*!< Offset: 0x028 (R/W) Configurable Fault Status Register */
/*/
    __IO uint32_t HFSR;           /*!< Offset: 0x02C (R/W) HardFault Status Register          */
    __IO uint32_t DFSR;           /*!< Offset: 0x030 (R/W) Debug Fault Status Register        */
    __IO uint32_t MMFAR;          /*!< Offset: 0x034 (R/W) MemManage Fault Address Register    */
/*/
    __IO uint32_t BFAR;           /*!< Offset: 0x038 (R/W) BusFault Address Register          */
    __IO uint32_t AFSR;           /*!< Offset: 0x03C (R/W) Auxiliary Fault Status Register    */
    __I uint32_t ID_PFR[2];       /*!< Offset: 0x040 (R/ ) Processor Feature Register          */
    __I uint32_t ID_DFR;          /*!< Offset: 0x048 (R/ ) Debug Feature Register            */
    __I uint32_t ID_AFR;          /*!< Offset: 0x04C (R/ ) Auxiliary Feature Register          */
    __I uint32_t ID_MFR[4];       /*!< Offset: 0x050 (R/ ) Memory Model Feature Register    */
/*/
    __I uint32_t ID_ISAR[5];      /*!< Offset: 0x060 (R/ ) Instruction Set Attributes Register
    uint32_t RESERVED0[1];
    __I uint32_t CLIDR;           /*!< Offset: 0x078 (R/ ) Cache Level ID register          */
*/
}

```

```

__I uint32_t CTR;           /*!< Offset: 0x07C (R/ ) Cache Type register          */
__I uint32_t CCSIDR;        /*!< Offset: 0x080 (R/ ) Cache Size ID Register      */
__IO uint32_t CSSELR;       /*!< Offset: 0x084 (R/W) Cache Size Selection Register */
__IO uint32_t CPACR;        /*!< Offset: 0x088 (R/W) Coprocessor Access Control Register */
*/
uint32_t RESERVED3[93];

__O uint32_t STIR;         /*!< Offset: 0x200 ( /W) Software Triggered Interrupt Register */
*/
uint32_t RESERVED4[15];

__I uint32_t MVFR0;        /*!< Offset: 0x240 (R/ ) Media and VFP Feature Register 0 */
*/
__I uint32_t MVFR1;        /*!< Offset: 0x244 (R/ ) Media and VFP Feature Register 1 */
*/
__I uint32_t MVFR2;        /*!< Offset: 0x248 (R/ ) Media and VFP Feature Register 1 */
*/
uint32_t RESERVED5[1];

__O uint32_t ICIALLU;      /*!< Offset: 0x250 ( /W) I-Cache Invalidate All to PoU      */
uint32_t RESERVED6[1];

__O uint32_t ICIMVAU;      /*!< Offset: 0x258 ( /W) I-Cache Invalidate by MVA to PoU */
*/
__O uint32_t DCIMVAC;      /*!< Offset: 0x25C ( /W) D-Cache Invalidate by MVA to PoC */
*/
__O uint32_t DCISW;         /*!< Offset: 0x260 ( /W) D-Cache Invalidate by Set-way */
*/
__O uint32_t DCCMVAU;      /*!< Offset: 0x264 ( /W) D-Cache Clean by MVA to PoU */
*/
__O uint32_t DCCMVAC;      /*!< Offset: 0x268 ( /W) D-Cache Clean by MVA to PoC */
*/
__O uint32_t DCCSW;         /*!< Offset: 0x26C ( /W) D-Cache Clean by Set-way          */
__O uint32_t DCCIMVAC;     /*!< Offset: 0x270 ( /W) D-Cache Clean and Invalidate by MVA to
PoC      */
*/
__O uint32_t DCCISW;        /*!< Offset: 0x274 ( /W) D-Cache Clean and Invalidate by Set-way */
*/

```

```

    uint32_t RESERVED7[6];

    __IO uint32_t ITCMCR;           /*!< Offset: 0x290 (R/W) Instruction Tightly-Coupled Memory Control
Register */

    __IO uint32_t DTCMCR;           /*!< Offset: 0x294 (R/W) Data Tightly-Coupled Memory Control
Registers */

    __IO uint32_t AHBPCR;           /*!< Offset: 0x298 (R/W) AHB Control Register */

    __IO uint32_t CACR;             /*!< Offset: 0x29C (R/W) L1 Cache Control Register */

    __IO uint32_t AHBSCR;           /*!< Offset: 0x2A0 (R/W) AHB Slave Control Register
*/
}

uint32_t RESERVED8[1];

__IO uint32_t ABFSR;             /*!< Offset: 0x2A8 (R/W) Auxiliary Bus Fault Status Register
*/
}

} SCB_Type;

```

```

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24          /*!< SCB CPUID:
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk      (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)  /*!<
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos         20          /*!< SCB CPUID: VARIANT Position
*/

#define SCB_CPUID_VARIANT_Msk         (0xFUL << SCB_CPUID_VARIANT_Pos)  /*!< SCB CPUID:
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos     16          /*!< SCB CPUID:
ARCHITECTURE Position */

#define SCB_CPUID_ARCHITECTURE_Msk     (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)  /*!<
SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos          4           /*!< SCB CPUID: PARTNO Position
*/

```

```

#define SCB_CPUID_PARTNO_Msk      (0xFFFFUL << SCB_CPUID_PARTNO_Pos)      /*!< SCB
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos    0                               /*!< SCB CPUID: REVISION Position
*/
#define SCB_CPUID_REVISION_Msk    (0xFUL /*<< SCB_CPUID_REVISION_Pos*/)      /*!< SCB
CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos   31                             /*!< SCB ICSR: NMIPENDSET
Position */
#define SCB_ICSR_NMIPENDSET_Msk   (1UL << SCB_ICSR_NMIPENDSET_Pos)      /*!< SCB ICSR:
NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos    28                             /*!< SCB ICSR: PENDSVSET
Position */
#define SCB_ICSR_PENDSVSET_Msk    (1UL << SCB_ICSR_PENDSVSET_Pos)      /*!< SCB ICSR:
PENDSVSET Mask */

#define SCB_ICSR_PENDSVCLR_Pos   27                             /*!< SCB ICSR: PENDSVCLR
Position */
#define SCB_ICSR_PENDSVCLR_Msk   (1UL << SCB_ICSR_PENDSVCLR_Pos)      /*!< SCB ICSR:
PENDSVCLR Mask */

#define SCB_ICSR_PENDSTSET_Pos   26                             /*!< SCB ICSR: PENDSTSET Position
*/
#define SCB_ICSR_PENDSTSET_Msk   (1UL << SCB_ICSR_PENDSTSET_Pos)      /*!< SCB ICSR:
PENDSTSET Mask */

#define SCB_ICSR_PENDSTCLR_Pos   25                             /*!< SCB ICSR: PENDSTCLR
Position */
#define SCB_ICSR_PENDSTCLR_Msk   (1UL << SCB_ICSR_PENDSTCLR_Pos)      /*!< SCB ICSR:
PENDSTCLR Mask */

```

```

#define SCB_ICSR_ISRPREEMPT_Pos      23          /*!< SCB ICSR: ISRPREEMPT
Position */

#define SCB_ICSR_ISRPREEMPT_Msk      (1UL << SCB_ICSR_ISRPREEMPT_Pos)    /*!< SCB ICSR:
ISRPREEMPT Mask */

#define SCB_ICSR_ISRPENDING_Pos     22          /*!< SCB ICSR: ISR PENDING
Position */

#define SCB_ICSR_ISRPENDING_Msk     (1UL << SCB_ICSR_ISRPENDING_Pos)    /*!< SCB ICSR:
ISR PENDING Mask */

#define SCB_ICSR_VECTPENDING_Pos    12          /*!< SCB ICSR: VECT PENDING
Position */

#define SCB_ICSR_VECTPENDING_Msk    (0x1FFUL << SCB_ICSR_VECTPENDING_Pos)  /*!< SCB
ICSR: VECT PENDING Mask */

#define SCB_ICSR_RETTOBASE_Pos      11          /*!< SCB ICSR: RET TO BASE
Position */

#define SCB_ICSR_RETTOBASE_Msk      (1UL << SCB_ICSR_RETTOBASE_Pos)    /*!< SCB ICSR:
RET TO BASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos     0           /*!< SCB ICSR: VECT ACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk     (0x1FFFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/) /*!< SCB
ICSR: VECT ACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */

#define SCB_VTOR_TBLOFF_Pos        7           /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk        (0xFFFFFFFFUL << SCB_VTOR_TBLOFF_Pos)  /*!< SCB
VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */

```

```

#define SCB_AIRCR_VECTKEY_Pos      16          /*!< SCB AIRCR: VECTKEY Position
*/
#define SCB_AIRCR_VECTKEY_Msk      (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)  /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos  16          /*!< SCB AIRCR: VECTKEYSTAT
Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk  (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos) /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos    15          /*!< SCB AIRCR: ENDIANESS
Position */
#define SCB_AIRCR_ENDIANESS_Msk    (1UL << SCB_AIRCR_ENDIANESS_Pos)  /*!< SCB
AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_PRIGROUP_Pos     8           /*!< SCB AIRCR: PRIGROUP
Position */
#define SCB_AIRCR_PRIGROUP_Msk     (7UL << SCB_AIRCR_PRIGROUP_Pos)  /*!< SCB
AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos   2           /*!< SCB AIRCR: SYSRESETREQ
Position */
#define SCB_AIRCR_SYSRESETREQ_Msk  (1UL << SCB_AIRCR_SYSRESETREQ_Pos) /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos 1           /*!< SCB AIRCR: VECTCLRACTIVE
Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos) /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

#define SCB_AIRCR_VECTRESET_Pos     0           /*!< SCB AIRCR: VECTRESET
Position */
#define SCB_AIRCR_VECTRESET_Msk    (1UL /*<< SCB_AIRCR_VECTRESET_Pos*/) /*!< SCB
AIRCR: VECTRESET Mask */

```

```

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos      4          /*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk      (1UL << SCB_SCR_SEVONPEND_Pos)    /*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos     2          /*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk      (1UL << SCB_SCR_SLEEPDEEP_Pos)    /*!< SCB SCR: SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos   1          /*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk    (1UL << SCB_SCR_SLEEPONEXIT_Pos)    /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_BP_Pos           18         /*!< SCB CCR: Branch prediction enable bit Position */
#define SCB_CCR_BP_Msk            (1UL << SCB_CCR_BP_Pos)        /*!< SCB CCR: Branch prediction enable bit Mask */

#define SCB_CCR_IC_Pos           17         /*!< SCB CCR: Instruction cache enable bit Position */
#define SCB_CCR_IC_Msk            (1UL << SCB_CCR_IC_Pos)        /*!< SCB CCR: Instruction cache enable bit Mask */

#define SCB_CCR_DC_Pos           16         /*!< SCB CCR: Cache enable bit Position */
#define SCB_CCR_DC_Msk            (1UL << SCB_CCR_DC_Pos)        /*!< SCB CCR: Cache enable bit Mask */

```

```

#define SCB_CCR_STKALIGN_Pos      9          /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk      (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR:
STKALIGN Mask */

#define SCB_CCR_BFHFMIGN_Pos      8          /*!< SCB CCR: BFHFMIGN
Position */
#define SCB_CCR_BFHFMIGN_Msk      (1UL << SCB_CCR_BFHFMIGN_Pos)      /*!< SCB CCR:
BFHFMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos      4          /*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk      (1UL << SCB_CCR_DIV_0_TRP_Pos)      /*!< SCB CCR:
DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos      3          /*!< SCB CCR: UNALIGN_TRP
Position */
#define SCB_CCR_UNALIGN_TRP_Msk      (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB
CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos      1          /*!< SCB CCR: USERSETMPEND
Position */
#define SCB_CCR_USERSETMPEND_Msk      (1UL << SCB_CCR_USERSETMPEND_Pos)      /*!< SCB
CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos      0          /*!< SCB CCR:
NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDENA_Msk      (1UL /*<< SCB_CCR_NONBASETHRDENA_Pos*/)      /*!< SCB CCR:
NONBASETHRDENA Mask */

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_USGFAULTENA_Pos      18         /*!< SCB SHCSR:
USGFAULTENA Position */

```

```
#define SCB_SHCSR_USGFAULTENA_Msk      (1UL << SCB_SHCSR_USGFAULTENA_Pos)      /*!< SCB
SHCSR: USGFAULTENA Mask */

#define SCB_SHCSR_BUSFAULTENA_Pos       17                                /*!< SCB SHCSR: BUSFAULTENA
Position */

#define SCB_SHCSR_BUSFAULTENA_Msk      (1UL << SCB_SHCSR_BUSFAULTENA_Pos)      /*!< SCB
SHCSR: BUSFAULTENA Mask */

#define SCB_SHCSR_MEMFAULTENA_Pos      16                                /*!< SCB SHCSR:
MEMFAULTENA Position */

#define SCB_SHCSR_MEMFAULTENA_Msk      (1UL << SCB_SHCSR_MEMFAULTENA_Pos)      /*!<
SCB SHCSR: MEMFAULTENA Mask */

#define SCB_SHCSR_SVCALLPENDED_Pos     15                                /*!< SCB SHCSR:
SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk     (1UL << SCB_SHCSR_SVCALLPENDED_Pos)      /*!<
SCB SHCSR: SVCALLPENDED Mask */

#define SCB_SHCSR_BUSFAULTPENDED_Pos   14                                /*!< SCB SHCSR:
BUSFAULTPENDED Position */

#define SCB_SHCSR_BUSFAULTPENDED_Msk   (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)      /*!<
SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos   13                                /*!< SCB SHCSR:
MEMFAULTPENDED Position */

#define SCB_SHCSR_MEMFAULTPENDED_Msk   (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)      /*!<
SCB SHCSR: MEMFAULTPENDED Mask */

#define SCB_SHCSR_USGFAULTPENDED_Pos   12                                /*!< SCB SHCSR:
USGFAULTPENDED Position */

#define SCB_SHCSR_USGFAULTPENDED_Msk   (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)      /*!<
SCB SHCSR: USGFAULTPENDED Mask */
```

#define SCB_SHCSR_SYSTICKACT_Pos	11	/*!< SCB SHCSR: SYSTICKACT Position */
#define SCB_SHCSR_SYSTICKACT_Msk	(1UL << SCB_SHCSR_SYSTICKACT_Pos)	/*!< SCB SHCSR: SYSTICKACT Mask */
#define SCB_SHCSR_PENDSVACT_Pos	10	/*!< SCB SHCSR: PENDSVACT Position */
#define SCB_SHCSR_PENDSVACT_Msk	(1UL << SCB_SHCSR_PENDSVACT_Pos)	/*!< SCB SHCSR: PENDSVACT Mask */
#define SCB_SHCSR_MONITORACT_Pos	8	/*!< SCB SHCSR: MONITORACT Position */
#define SCB_SHCSR_MONITORACT_Msk	(1UL << SCB_SHCSR_MONITORACT_Pos)	/*!< SCB SHCSR: MONITORACT Mask */
#define SCB_SHCSR_SVCALLACT_Pos	7	/*!< SCB SHCSR: SVCALLACT Position */
#define SCB_SHCSR_SVCALLACT_Msk	(1UL << SCB_SHCSR_SVCALLACT_Pos)	/*!< SCB SHCSR: SVCALLACT Mask */
#define SCB_SHCSR_USGFAULTACT_Pos	3	/*!< SCB SHCSR: USGFAULTACT Position */
#define SCB_SHCSR_USGFAULTACT_Msk	(1UL << SCB_SHCSR_USGFAULTACT_Pos)	/*!< SCB SHCSR: USGFAULTACT Mask */
#define SCB_SHCSR_BUSFAULTACT_Pos	1	/*!< SCB SHCSR: BUSFAULTACT Position */
#define SCB_SHCSR_BUSFAULTACT_Msk	(1UL << SCB_SHCSR_BUSFAULTACT_Pos)	/*!< SCB SHCSR: BUSFAULTACT Mask */
#define SCB_SHCSR_MEMFAULTACT_Pos	0	/*!< SCB SHCSR: MEMFAULTACT Position */
#define SCB_SHCSR_MEMFAULTACT_Msk	(1UL /*<< SCB_SHCSR_MEMFAULTACT_Pos*/)	/*!< SCB SHCSR: MEMFAULTACT Mask */

```

/* SCB Configurable Fault Status Registers Definitions */

#define SCB_CFSR_USGFAULTSR_Pos      16          /*!< SCB CFSR: Usage Fault
Status Register Position */

#define SCB_CFSR_USGFAULTSR_Msk     (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos)  /*!< SCB
CFSR: Usage Fault Status Register Mask */

#define SCB_CFSR_BUSFAULTSR_Pos     8           /*!< SCB CFSR: Bus Fault Status
Register Position */

#define SCB_CFSR_BUSFAULTSR_Msk    (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos)  /*!< SCB
CFSR: Bus Fault Status Register Mask */

#define SCB_CFSR_MEMFAULTSR_Pos     0           /*!< SCB CFSR: Memory Manage
Fault Status Register Position */

#define SCB_CFSR_MEMFAULTSR_Msk    (0xFFUL /*<< SCB_CFSR_MEMFAULTSR_Pos*/)  /*!<
SCB CFSR: Memory Manage Fault Status Register Mask */

/* SCB Hard Fault Status Registers Definitions */

#define SCB_HFSR_DEBUGEVT_Pos       31          /*!< SCB HFSR: DEBUGEVT
Position */

#define SCB_HFSR_DEBUGEVT_Msk      (1UL << SCB_HFSR_DEBUGEVT_Pos)  /*!< SCB HFSR:
DEBUGEVT Mask */

#define SCB_HFSR_FORCED_Pos        30          /*!< SCB HFSR: FORCED Position */

#define SCB_HFSR_FORCED_Msk       (1UL << SCB_HFSR_FORCED_Pos)  /*!< SCB HFSR:
FORCED Mask */

#define SCB_HFSR_VECTTBL_Pos        1           /*!< SCB HFSR: VECTTBL Position */

#define SCB_HFSR_VECTTBL_Msk      (1UL << SCB_HFSR_VECTTBL_Pos)  /*!< SCB HFSR:
VECTTBL Mask */

/* SCB Debug Fault Status Register Definitions */

```

```

#define SCB_DFSR_EXTERNAL_Pos      4          /*!< SCB DFSR: EXTERNAL Position */
*/
#define SCB_DFSR_EXTERNAL_Msk     (1UL << SCB_DFSR_EXTERNAL_Pos)    /*!< SCB DFSR:
EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos       3          /*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk      (1UL << SCB_DFSR_VCATCH_Pos)    /*!< SCB DFSR:
VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos     2          /*!< SCB DFSR: DWTTRAP Position */
*/
#define SCB_DFSR_DWTTRAP_Msk    (1UL << SCB_DFSR_DWTTRAP_Pos)    /*!< SCB DFSR:
DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos        1          /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk       (1UL << SCB_DFSR_BKPT_Pos)    /*!< SCB DFSR: BKPT
Mask */

#define SCB_DFSR_HALTED_Pos      0          /*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk     (1UL /*<< SCB_DFSR_HALTED_Pos*/) /*!< SCB DFSR:
HALTED Mask */

/* Cache Level ID register */
#define SCB_CLIDR_LOUU_Pos       27         /*!< SCB CLIDR: LoUU Position */
#define SCB_CLIDR_LOUU_Msk      (7UL << SCB_CLIDR_LOUU_Pos)    /*!< SCB CLIDR:
LoUU Mask */

#define SCB_CLIDR_LOC_Pos        24         /*!< SCB CLIDR: LoC Position */
#define SCB_CLIDR_LOC_Msk       (7UL << SCB_CLIDR_FORMAT_Pos) /*!< SCB CLIDR: LoC
Mask */

/* Cache Type register */

```

```

#define SCB_CTR_FORMAT_Pos      29           /*!< SCB CTR: Format Position */
#define SCB_CTR_FORMAT_Msk     (7UL << SCB_CTR_FORMAT_Pos)    /*!< SCB CTR:
Format Mask */

#define SCB_CTR_CWG_Pos        24           /*!< SCB CTR: CWG Position */
#define SCB_CTR_CWG_Msk       (0xFUL << SCB_CTR_CWG_Pos)    /*!< SCB CTR: CWG
Mask */

#define SCB_CTR_ERG_Pos        20           /*!< SCB CTR: ERG Position */
#define SCB_CTR_ERG_Msk       (0xFUL << SCB_CTR_ERG_Pos)    /*!< SCB CTR: ERG
Mask */

#define SCB_CTR_DMINLINE_Pos   16           /*!< SCB CTR: DminLine Position */
#define SCB_CTR_DMINLINE_Msk  (0xFUL << SCB_CTR_DMINLINE_Pos) /*!< SCB CTR:
DminLine Mask */

#define SCB_CTR_IMINLINE_Pos   0            /*!< SCB CTR: ImInLine Position */
#define SCB_CTR_IMINLINE_Msk  (0xFUL /*<< SCB_CTR_IMINLINE_Pos*/) /*!< SCB CTR:
ImInLine Mask */

/* Cache Size ID Register */

#define SCB_CCSIDR_WT_Pos      31           /*!< SCB CCSIDR: WT Position */
#define SCB_CCSIDR_WT_Msk     (7UL << SCB_CCSIDR_WT_Pos)    /*!< SCB CCSIDR: WT
Mask */

#define SCB_CCSIDR_WB_Pos      30           /*!< SCB CCSIDR: WB Position */
#define SCB_CCSIDR_WB_Msk     (7UL << SCB_CCSIDR_WB_Pos)    /*!< SCB CCSIDR: WB
Mask */

#define SCB_CCSIDR_RA_Pos      29           /*!< SCB CCSIDR: RA Position */
#define SCB_CCSIDR_RA_Msk     (7UL << SCB_CCSIDR_RA_Pos)    /*!< SCB CCSIDR: RA
Mask */

```

```

#define SCB_CCSIDR_WA_Pos           28          /*!< SCB CCSIDR: WA Position */
#define SCB_CCSIDR_WA_Msk          (7UL << SCB_CCSIDR_WA_Pos)      /*!< SCB CCSIDR: WA
Mask */

#define SCB_CCSIDR_NUMSETS_Pos     13          /*!< SCB CCSIDR: NumSets
Position */
#define SCB_CCSIDR_NUMSETS_Msk    (0x7FFFUL << SCB_CCSIDR_NUMSETS_Pos)  /*!< SCB
CCSIDR: NumSets Mask */

#define SCB_CCSIDR_ASSOCIATIVITY_Pos 3          /*!< SCB CCSIDR: Associativity
Position */
#define SCB_CCSIDR_ASSOCIATIVITY_Msk (0x3FFUL << SCB_CCSIDR_ASSOCIATIVITY_Pos) /*!< SCB
CCSIDR: Associativity Mask */

#define SCB_CCSIDR_LINESIZE_Pos    0          /*!< SCB CCSIDR: LineSize Position
*/
#define SCB_CCSIDR_LINESIZE_Msk   (7UL /*<< SCB_CCSIDR_LINESIZE_Pos*/)  /*!< SCB
CCSIDR: LineSize Mask */

/* Cache Size Selection Register */
#define SCB_CSSELR_LEVEL_Pos       1          /*!< SCB CSSELR: Level Position */
#define SCB_CSSELR_LEVEL_Msk      (7UL << SCB_CSSELR_LEVEL_Pos)      /*!< SCB CSSELR:
Level Mask */

#define SCB_CSSELR_IND_Pos        0          /*!< SCB CSSELR: InD Position */
#define SCB_CSSELR_IND_Msk       (1UL /*<< SCB_CSSELR_IND_Pos*/)  /*!< SCB CSSELR:
InD Mask */

/* SCB Software Triggered Interrupt Register */
#define SCB_STIR_INTID_Pos        0          /*!< SCB STIR: INTID Position */

```

```

#define SCB_STIR_INTID_Msk      (0x1FFUL /*<< SCB_STIR_INTID_Pos*/)      /*!< SCB STIR: INTID
Mask */

/* Instruction Tightly-Coupled Memory Control Register*/

#define SCB_ITCMCR_SZ_Pos      3                                /*!< SCB ITCMCR: SZ Position */
#define SCB_ITCMCR_SZ_Msk      (0xFUL << SCB_ITCMCR_SZ_Pos)      /*!< SCB ITCMCR: SZ
Mask */

#define SCB_ITCMCR_RETEN_Pos    2                                /*!< SCB ITCMCR: RETEN Position
*/
#define SCB_ITCMCR_RETEN_Msk    (1UL << SCB_ITCMCR_RETEN_Pos)      /*!< SCB ITCMCR:
RETEN Mask */

#define SCB_ITCMCR_RMW_Pos     1                                /*!< SCB ITCMCR: RMW Position */
#define SCB_ITCMCR_RMW_Msk     (1UL << SCB_ITCMCR_RMW_Pos)      /*!< SCB
ITCMCR: RMW Mask */

#define SCB_ITCMCR_EN_Pos      0                                /*!< SCB ITCMCR: EN Position */
#define SCB_ITCMCR_EN_Msk      (1UL /*<< SCB_ITCMCR_EN_Pos*/)      /*!< SCB ITCMCR:
EN Mask */

/* Data Tightly-Coupled Memory Control Registers*/

#define SCB_DTCMCR_SZ_Pos      3                                /*!< SCB DTCMCR: SZ Position */
#define SCB_DTCMCR_SZ_Msk      (0xFUL << SCB_DTCMCR_SZ_Pos)      /*!< SCB DTCMCR:
SZ Mask */

#define SCB_DTCMCR_RETEN_Pos    2                                /*!< SCB DTCMCR: RETEN Position
*/
#define SCB_DTCMCR_RETEN_Msk    (1UL << SCB_DTCMCR_RETEN_Pos)      /*!< SCB
DTCMCR: RETEN Mask */

```

```

#define SCB_DTCMCR_RMW_Pos      1          /*!< SCB DTCMCR: RMW Position */
*/
#define SCB_DTCMCR_RMW_Msk      (1UL << SCB_DTCMCR_RMW_Pos)      /*!< SCB
DTCMCR: RMW Mask */

#define SCB_DTCMCR_EN_Pos       0          /*!< SCB DTCMCR: EN Position */
#define SCB_DTCMCR_EN_Msk       (1UL /*<< SCB_DTCMCR_EN_Pos*/)      /*!< SCB
DTCMCR: EN Mask */

/* AHB Control Register */
#define SCB_AHBPCR_SZ_Pos       1          /*!< SCB AHBPCR: SZ Position */
#define SCB_AHBPCR_SZ_Msk       (7UL << SCB_AHBPCR_SZ_Pos)      /*!< SCB AHBPCR: SZ
Mask */

#define SCB_AHBPCR_EN_Pos       0          /*!< SCB AHBPCR: EN Position */
#define SCB_AHBPCR_EN_Msk       (1UL /*<< SCB_AHBPCR_EN_Pos*/)      /*!< SCB AHBPCR:
EN Mask */

/* L1 Cache Control Register */
#define SCB_CACR_FORCEWT_Pos    2          /*!< SCB CACR: FORCEWT Position */
*/
#define SCB_CACR_FORCEWT_Msk    (1UL << SCB_CACR_FORCEWT_Pos)      /*!< SCB CACR:
FORCEWT Mask */

#define SCB_CACR_ECCEN_Pos      1          /*!< SCB CACR: ECCEN Position */
#define SCB_CACR_ECCEN_Msk      (1UL << SCB_CACR_ECCEN_Pos)      /*!< SCB CACR:
ECCEN Mask */

#define SCB_CACR_SIWT_Pos       0          /*!< SCB CACR: SIWT Position */
#define SCB_CACR_SIWT_Msk       (1UL /*<< SCB_CACR_SIWT_Pos*/)      /*!< SCB CACR:
SIWT Mask */

```

```

/* AHBS control register */

#define SCB_AHBSCR_INITCOUNT_Pos      11           /*!< SCB AHBSCR: INITCOUNT Position */
#define SCB_AHBSCR_INITCOUNT_Msk     (0x1FUL << SCB_AHBPCR_INITCOUNT_Pos)    /*!< SCB AHBSCR: INITCOUNT Mask */

#define SCB_AHBSCR_TPRI_Pos         2            /*!< SCB AHBSCR: TPRI Position */
#define SCB_AHBSCR_TPRI_Msk        (0x1FFUL << SCB_AHBPCR_TPRI_Pos)    /*!< SCB AHBSCR: TPRI Mask */

#define SCB_AHBSCR_CTL_Pos          0            /*!< SCB AHBSCR: CTL Position*/
#define SCB_AHBSCR_CTL_Msk         (3UL /*<< SCB_AHBPCR_CTL_Pos*/)    /*!< SCB AHBSCR: CTL Mask */

/* Auxiliary Bus Fault Status Register */

#define SCB_ABFSR_AXIMTYPE_Pos      8           /*!< SCB ABFSR: AXIMTYPE Position*/
#define SCB_ABFSR_AXIMTYPE_Msk     (3UL << SCB_ABFSR_AXIMTYPE_Pos)    /*!< SCB ABFSR: AXIMTYPE Mask */

#define SCB_ABFSR_EPPB_Pos          4            /*!< SCB ABFSR: EPPB Position*/
#define SCB_ABFSR_EPPB_Msk         (1UL << SCB_ABFSR_EPPB_Pos)    /*!< SCB ABFSR: EPPB Mask */

#define SCB_ABFSR_AXIM_Pos          3            /*!< SCB ABFSR: AXIM Position*/
#define SCB_ABFSR_AXIM_Msk         (1UL << SCB_ABFSR_AXIM_Pos)    /*!< SCB ABFSR: AXIM Mask */

#define SCB_ABFSR_AHBP_Pos          2            /*!< SCB ABFSR: AHBP Position*/
#define SCB_ABFSR_AHBP_Msk         (1UL << SCB_ABFSR_AHBP_Pos)    /*!< SCB ABFSR: AHBP Mask */

```

```

#define SCB_ABFSR_DTCM_Pos      1                  /*!< SCB ABFSR: DTCM Position*/
#define SCB_ABFSR_DTCM_Msk      (1UL << SCB_ABFSR_DTCM_Pos)    /*!< SCB ABFSR:
DTCM Mask */

#define SCB_ABFSR_ITCM_Pos      0                  /*!< SCB ABFSR: ITCM Position*/
#define SCB_ABFSR_ITCM_Msk      (1UL /*<< SCB_ABFSR_ITCM_Pos*/) /*!< SCB ABFSR:
ITCM Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
\brief Type definitions for the System Control and ID Register not in the SCB
@{
*/

```

```

/** \brief Structure type to access the System Control and ID Register not in the SCB.
*/
typedef struct
{
    uint32_t RESERVED0[1];
    __I uint32_t ICTR;           /*!< Offset: 0x004 (R/ ) Interrupt Controller Type Register */
    __IO uint32_t ACTLR;        /*!< Offset: 0x008 (R/W) Auxiliary Control Register */
} SCnSCB_Type;

```

```

/* Interrupt Controller Type Register Definitions */

#define SCnSCB_ICTR_INTLINESNUM_Pos 0          /*!< ICTR: INTLINESNUM Position */
*/
```

```

#define SCnSCB_ICTR_INTLINESNUM_Msk      (0xFUL /*<< SCnSCB_ICTR_INTLINESNUM_Pos*/) /*!<
ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define SCnSCB_ACTLR_DISITMATBFLUSH_Pos  12          /*!< ACTLR: DISITMATBFLUSH
Position */

#define SCnSCB_ACTLR_DISITMATBFLUSH_Msk  (1UL << SCnSCB_ACTLR_DISITMATBFLUSH_Pos) /*!<
ACTLR: DISITMATBFLUSH Mask */

#define SCnSCB_ACTLR_DISRAMODE_Pos     11          /*!< ACTLR: DISRAMODE
Position */

#define SCnSCB_ACTLR_DISRAMODE_Msk    (1UL << SCnSCB_ACTLR_DISRAMODE_Pos) /*!<
ACTLR: DISRAMODE Mask */

#define SCnSCB_ACTLR_FPEXCODIS_Pos   10          /*!< ACTLR: FPEXCODIS Position */
*/

#define SCnSCB_ACTLR_FPEXCODIS_Msk  (1UL << SCnSCB_ACTLR_FPEXCODIS_Pos) /*!< ACTLR:
FPEXCODIS Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos     2           /*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk   (1UL << SCnSCB_ACTLR_DISFOLD_Pos) /*!< ACTLR:
DISFOLD Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos  0           /*!< ACTLR: DISMCYCINT
Position */

#define SCnSCB_ACTLR_DISMCYCINT_Msk (1UL /*<< SCnSCB_ACTLR_DISMCYCINT_Pos*/) /*!<
ACTLR: DISMCYCINT Mask */

/*@} end of group CMSIS_SCnotSCB */

```

```
/** \ingroup CMSIS_core_register
```

```

\defgroup CMSIS_SysTick System Tick Timer (SysTick)

\brief Type definitions for the System Timer Registers.

@{

*


/** \brief Structure type to access the System Timer (SysTick).

*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I  uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
} SysTick_Type;

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos     16          /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk    (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2           /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk    (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos      1           /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk     (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

```

```

#define SysTick_CTRL_ENABLE_Pos      0           /*!< SysTick CTRL: ENABLE Position */
/*/

#define SysTick_CTRL_ENABLE_Msk     (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)    /*!< SysTick
CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos     0           /*!< SysTick LOAD: RELOAD
Position */

#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/) /*!<
SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos     0           /*!< SysTick VAL: CURRENT
Position */

#define SysTick_VAL_CURRENT_Msk     (0xFFFFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/) /*!<
SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos     31          /*!< SysTick CALIB: NOREF Position */
/*/

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)        /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos       30          /*!< SysTick CALIB: SKEW Position */
/*/

#define SysTick_CALIB_SKEW_Msk       (1UL << SysTick_CALIB_SKEW_Pos)        /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos      0           /*!< SysTick CALIB: TENMS Position */
/*/

#define SysTick_CALIB_TENMS_Msk     (0xFFFFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

```

```

/*@} end of group CMSIS_SysTick */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_ITM Instrumentation Trace Macrocell (ITM)
 \brief Type definitions for the Instrumentation Trace Macrocell (ITM)

 @{
 */

/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
 */
typedef struct
{
    __O union
    {
        __O uint8_t u8;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit      */
        __O uint16_t u16;        /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit     */
        __O uint32_t u32;        /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit     */
    } PORT [32];             /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers   */

    uint32_t RESERVED0[864];

    __IO uint32_t TER;        /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register    */
    uint32_t RESERVED1[15];

    __IO uint32_t TPR;        /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register */
    uint32_t RESERVED2[15];

    __IO uint32_t TCR;        /*!< Offset: 0xE80 (R/W) ITM Trace Control Register   */
    uint32_t RESERVED3[29];

    __O uint32_t IWR;         /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register */
    __I uint32_t IRR;         /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register */
    __IO uint32_t IMCR;       /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
}

```

```

    uint32_t RESERVED4[43];

    __O uint32_t LAR;           /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register      */
    __I uint32_t LSR;           /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register       */
    uint32_t RESERVED5[6];

    __I uint32_t PID4;          /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
    __I uint32_t PID5;          /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
    __I uint32_t PID6;          /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
    __I uint32_t PID7;          /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
    __I uint32_t PID0;          /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
    __I uint32_t PID1;          /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
    __I uint32_t PID2;          /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
    __I uint32_t PID3;          /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
    __I uint32_t CID0;          /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
    __I uint32_t CID1;          /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
    __I uint32_t CID2;          /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */
    __I uint32_t CID3;          /*!< Offset: 0xFFC (R/ ) ITM Component Identification Register #3 */

} ITM_Type;

```

```

/* ITM Trace Privilege Register Definitions */

#define ITM_TPR_PRIVMASK_Pos      0                  /*!< ITM TPR: PRIVMASK Position */
#define ITM_TPR_PRIVMASK_Msk      (0xFUL /*<< ITM_TPR_PRIVMASK_Pos*/)      /*!< ITM TPR: PRIVMASK Mask */

```

```

/* ITM Trace Control Register Definitions */

#define ITM_TCR_BUSY_Pos         23                /*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk         (1UL << ITM_TCR_BUSY_Pos)        /*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos   16                /*!< ITM TCR: ATBID Position */

```

#define ITM_TCR_TraceBusID_Msk	(0x7FUL << ITM_TCR_TraceBusID_Pos)	/*!< ITM TCR: ATBID Mask */
#define ITM_TCR_GTSFREQ_Pos	10	/*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk	(3UL << ITM_TCR_GTSFREQ_Pos)	/*!< ITM TCR: Global timestamp frequency Mask */
#define ITM_TCR_TSPrescale_Pos	8	/*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk	(3UL << ITM_TCR_TSPrescale_Pos)	/*!< ITM TCR: TSPrescale Mask */
#define ITM_TCR_SWOENA_Pos	4	/*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk	(1UL << ITM_TCR_SWOENA_Pos)	/*!< ITM TCR: SWOENA Mask */
#define ITM_TCR_DWTENA_Pos	3	/*!< ITM TCR: DWTENA Position */
#define ITM_TCR_DWTENA_Msk	(1UL << ITM_TCR_DWTENA_Pos)	/*!< ITM TCR: DWTENA Mask */
#define ITM_TCR_SYNCENA_Pos	2	/*!< ITM TCR: SYNCENA Position */
#define ITM_TCR_SYNCENA_Msk	(1UL << ITM_TCR_SYNCENA_Pos)	/*!< ITM TCR: SYNCENA Mask */
#define ITM_TCR_TSENA_Pos	1	/*!< ITM TCR: TSENA Position */
#define ITM_TCR_TSENA_Msk	(1UL << ITM_TCR_TSENA_Pos)	/*!< ITM TCR: TSENA Mask */
#define ITM_TCR_ITMENA_Pos	0	/*!< ITM TCR: ITM Enable bit Position */
#define ITM_TCR_ITMENA_Msk	(1UL /*<< ITM_TCR_ITMENA_Pos*/)	/*!< ITM TCR: ITM Enable bit Mask */

```

/* ITM Integration Write Register Definitions */

#define ITM_IWR_ATVALIDDM_Pos      0           /*!< ITM IWR: ATVALIDDM Position */
/*/

#define ITM_IWR_ATVALIDDM_Msk     (1UL /*<< ITM_IWR_ATVALIDDM_Pos*/)      /*!< ITM IWR:
ATVALIDDM Mask */

/* ITM Integration Read Register Definitions */

#define ITMIRR_ATREADYM_Pos       0           /*!< ITM IRR: ATREADYM Position */
/*/

#define ITMIRR_ATREADYM_Msk      (1UL /*<< ITMIRR_ATREADYM_Pos*/)      /*!< ITM IRR:
ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */

#define ITMIMCR_INTEGRATION_Pos   0           /*!< ITM IMCR: INTEGRATION
Position */

#define ITMIMCR_INTEGRATION_Msk   (1UL /*<< ITMIMCR_INTEGRATION_Pos*/)      /*!< ITM
IMCR: INTEGRATION Mask */

/* ITM Lock Status Register Definitions */

#define ITMLSR_BytAcc_Pos         2           /*!< ITM LSR: ByteAcc Position */
#define ITMLSR_BytAcc_Msk         (1UL << ITMLSR_BytAcc_Pos)      /*!< ITM LSR: ByteAcc
Mask */

#define ITMLSR_Access_Pos         1           /*!< ITM LSR: Access Position */
#define ITMLSR_Access_Msk         (1UL << ITMLSR_Access_Pos)      /*!< ITM LSR: Access
Mask */

#define ITMLSR_Present_Pos        0           /*!< ITM LSR: Present Position */
#define ITMLSR_Present_Msk        (1UL /*<< ITMLSR_Present_Pos*/)      /*!< ITM LSR:
Present Mask */

```

```

/*}@*/ /* end of group CMSIS_ITM */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
\brief Type definitions for the Data Watchpoint and Trace (DWT)

@{
*/

/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).
*/
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) Control Register */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle Count Register */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI Count Register */
    __IO uint32_t EXCCNT;        /*!< Offset: 0x00C (R/W) Exception Overhead Count Register */
    __IO uint32_t SLEEPCNT;      /*!< Offset: 0x010 (R/W) Sleep Count Register */
    __IO uint32_t LSUCNT;        /*!< Offset: 0x014 (R/W) LSU Count Register */
    __IO uint32_t FOLDCNT;       /*!< Offset: 0x018 (R/W) Folded-instruction Count Register */
    __I uint32_t PCSR;          /*!< Offset: 0x01C (R/ ) Program Counter Sample Register */
    __IO uint32_t COMPO;         /*!< Offset: 0x020 (R/W) Comparator Register 0 */
    __IO uint32_t MASK0;         /*!< Offset: 0x024 (R/W) Mask Register 0 */
    __IO uint32_t FUNCTION0;     /*!< Offset: 0x028 (R/W) Function Register 0 */
    uint32_t RESERVED0[1];
    __IO uint32_t COMP1;         /*!< Offset: 0x030 (R/W) Comparator Register 1 */
    __IO uint32_t MASK1;         /*!< Offset: 0x034 (R/W) Mask Register 1 */
    __IO uint32_t FUNCTION1;     /*!< Offset: 0x038 (R/W) Function Register 1 */
    uint32_t RESERVED1[1];
}

```

```

__IO uint32_t COMP2;           /*!< Offset: 0x040 (R/W) Comparator Register 2      */
__IO uint32_t MASK2;          /*!< Offset: 0x044 (R/W) Mask Register 2           */
__IO uint32_t FUNCTION2;       /*!< Offset: 0x048 (R/W) Function Register 2        */
uint32_t RESERVED2[1];

__IO uint32_t COMP3;           /*!< Offset: 0x050 (R/W) Comparator Register 3      */
__IO uint32_t MASK3;          /*!< Offset: 0x054 (R/W) Mask Register 3           */
__IO uint32_t FUNCTION3;       /*!< Offset: 0x058 (R/W) Function Register 3        */
uint32_t RESERVED3[981];

__O uint32_t LAR;             /*!< Offset: 0xFB0 ( W) Lock Access Register       */
__I uint32_t LSR;             /*!< Offset: 0xFB4 ( R ) Lock Status Register      */

} DWT_Type;

/* DWT Control Register Definitions */

#define DWT_CTRL_NUMCOMP_Pos     28                  /*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk     (0xFUL << DWT_CTRL_NUMCOMP_Pos)    /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos    27                  /*!< DWT CTRL: NOTRCPKT Position */
#define DWT_CTRL_NOTRCPKT_Msk    (0x1UL << DWT_CTRL_NOTRCPKT_Pos)   /*!< DWT CTRL: NOTRCPKT Mask */

#define DWT_CTRL_NOEXTTRIG_Pos   26                  /*!< DWT CTRL: NOEXTTRIG Position */
#define DWT_CTRL_NOEXTTRIG_Msk   (0x1UL << DWT_CTRL_NOEXTTRIG_Pos)  /*!< DWT CTRL: NOEXTTRIG Mask */

#define DWT_CTRL_NOCYCCNT_Pos    25                  /*!< DWT CTRL: NOCYCCNT Position */
#define DWT_CTRL_NOCYCCNT_Msk    (0x1UL << DWT_CTRL_NOCYCCNT_Pos)  /*!< DWT CTRL: NOCYCCNT Mask */

```

#define DWT_CTRL_NOPRFCNT_Pos	24	/*!< DWT CTRL: NOPRFCNT Position */
#define DWT_CTRL_NOPRFCNT_Msk	(0x1UL << DWT_CTRL_NOPRFCNT_Pos)	/*!< DWT CTRL: NOPRFCNT Mask */
#define DWT_CTRL_CYCEVTENA_Pos	22	/*!< DWT CTRL: CYCEVTENA Position */
#define DWT_CTRL_CYCEVTENA_Msk	(0x1UL << DWT_CTRL_CYCEVTENA_Pos)	/*!< DWT CTRL: CYCEVTENA Mask */
#define DWT_CTRL_FOLDEVTENA_Pos	21	/*!< DWT CTRL: FOLDEVTENA Position */
#define DWT_CTRL_FOLDEVTENA_Msk	(0x1UL << DWT_CTRL_FOLDEVTENA_Pos)	/*!< DWT CTRL: FOLDEVTENA Mask */
#define DWT_CTRL_LSUEVTENA_Pos	20	/*!< DWT CTRL: LSUEVTENA Position */
#define DWT_CTRL_LSUEVTENA_Msk	(0x1UL << DWT_CTRL_LSUEVTENA_Pos)	/*!< DWT CTRL: LSUEVTENA Mask */
#define DWT_CTRL_SLEEPEVTENA_Pos	19	/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk	(0x1UL << DWT_CTRL_SLEEPEVTENA_Pos)	/*!< DWT CTRL: SLEEPEVTENA Mask */
#define DWT_CTRL_EXCEVTENA_Pos	18	/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk	(0x1UL << DWT_CTRL_EXCEVTENA_Pos)	/*!< DWT CTRL: EXCEVTENA Mask */
#define DWT_CTRL_CPIEVTEA_Pos	17	/*!< DWT CTRL: CPIEVTEA Position */

#define DWT_CTRL_CPIEVTENA_Msk	(0x1UL << DWT_CTRL_CPIEVTENA_Pos)	/*!< DWT CTRL: CPIEVTENA Mask */
#define DWT_CTRL_EXCTRCENA_Pos	16	/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk	(0x1UL << DWT_CTRL_EXCTRCENA_Pos)	/*!< DWT CTRL: EXCTRCENA Mask */
#define DWT_CTRL_PCSAMPLENA_Pos	12	/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk	(0x1UL << DWT_CTRL_PCSAMPLENA_Pos)	/*!< DWT CTRL: PCSAMPLENA Mask */
#define DWT_CTRL_SYNCTAP_Pos	10	/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk	(0x3UL << DWT_CTRL_SYNCTAP_Pos)	/*!< DWT CTRL: SYNCTAP Mask */
#define DWT_CTRL_CYCTAP_Pos	9	/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk	(0x1UL << DWT_CTRL_CYCTAP_Pos)	/*!< DWT CTRL: CYCTAP Mask */
#define DWT_CTRL_POSTINIT_Pos	5	/*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk	(0xFUL << DWT_CTRL_POSTINIT_Pos)	/*!< DWT CTRL: POSTINIT Mask */
#define DWT_CTRL_POSTRESET_Pos	1	/*!< DWT CTRL: POSTRESET Position */
#define DWT_CTRL_POSTRESET_Msk	(0xFUL << DWT_CTRL_POSTRESET_Pos)	/*!< DWT CTRL: POSTRESET Mask */
#define DWT_CTRL_CYCCNTENA_Pos	0	/*!< DWT CTRL: CYCCNTENA Position */

```

#define DWT_CTRL_CYCCNTENA_Msk      (0x1UL /*<< DWT_CTRL_CYCCNTENA_Pos*/) /*!< DWT
CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */

#define DWT_CPICNT_CPICNT_Pos      0                      /*!< DWT CPICNT: CPICNT Position
*/
#define DWT_CPICNT_CPICNT_Msk      (0xFFUL /*<< DWT_CPICNT_CPICNT_Pos*/) /*!< DWT
CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */

#define DWT_EXCCNT_EXCCNT_Pos      0                      /*!< DWT EXCCNT: EXCCNT
Position */
#define DWT_EXCCNT_EXCCNT_Msk      (0xFFUL /*<< DWT_EXCCNT_EXCCNT_Pos*/) /*!< DWT
EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */

#define DWT_SLEEPCNT_SLEEPCNT_Pos  0                      /*!< DWT SLEEPCNT: SLEEPCNT
Position */
#define DWT_SLEEPCNT_SLEEPCNT_Msk  (0xFFUL /*<< DWT_SLEEPCNT_SLEEPCNT_Pos*/) /*!<
DWT SLEEPCNT: SLEEPCNT Mask */

/* DWT LSU Count Register Definitions */

#define DWT_LSUCNT_LSUCNT_Pos      0                      /*!< DWT LSUCNT: LSUCNT Position
*/
#define DWT_LSUCNT_LSUCNT_Msk      (0xFFUL /*<< DWT_LSUCNT_LSUCNT_Pos*/) /*!< DWT
LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */

#define DWT_FOLDCNT_FOLDCNT_Pos   0                      /*!< DWT FOLDCNT: FOLDCNT
Position */
#define DWT_FOLDCNT_FOLDCNT_Msk   (0xFFUL /*<< DWT_FOLDCNT_FOLDCNT_Pos*/) /*!<
DWT FOLDCNT: FOLDCNT Mask */

```

```

/* DWT Comparator Mask Register Definitions */

#define DWT_MASK_MASK_Pos      0           /*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk      (0x1FUL /*<< DWT_MASK_MASK_Pos*/)    /*!< DWT
MASK: MASK Mask */

/* DWT Comparator Function Register Definitions */

#define DWT_FUNCTION_MATCHED_Pos   24          /*!< DWT FUNCTION:
MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk   (0x1UL << DWT_FUNCTION_MATCHED_Pos) /*!<
DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16          /*!< DWT FUNCTION:
DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL << DWT_FUNCTION_DATAVADDR1_Pos) /*!<
DWT FUNCTION: DATAVADDR1 Mask */

#define DWT_FUNCTION_DATAVADDR0_Pos 12          /*!< DWT FUNCTION:
DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk (0xFUL << DWT_FUNCTION_DATAVADDR0_Pos) /*!<
DWT FUNCTION: DATAVADDR0 Mask */

#define DWT_FUNCTION_DATAVSIZE_Pos 10          /*!< DWT FUNCTION:
DATAVSIZE Position */
#define DWT_FUNCTION_DATAVSIZE_Msk (0x3UL << DWT_FUNCTION_DATAVSIZE_Pos) /*!<
DWT FUNCTION: DATAVSIZE Mask */

#define DWT_FUNCTION_LNK1ENA_Pos   9           /*!< DWT FUNCTION: LNK1ENA
Position */
#define DWT_FUNCTION_LNK1ENA_Msk   (0x1UL << DWT_FUNCTION_LNK1ENA_Pos) /*!<
DWT FUNCTION: LNK1ENA Mask */

#define DWT_FUNCTION_DATAVMATCH_Pos 8           /*!< DWT FUNCTION:
DATAVMATCH Position */

```

```

#define DWT_FUNCTION_DATAVMATCH_Msk      (0x1UL << DWT_FUNCTION_DATAVMATCH_Pos)      /*!< DWT FUNCTION: DATAVMATCH Mask */

#define DWT_FUNCTION_CYCMATCH_Pos        7                                         /*!< DWT FUNCTION: CYCMATCH Position */

#define DWT_FUNCTION_CYCMATCH_Msk       (0x1UL << DWT_FUNCTION_CYCMATCH_Pos)      /*!< DWT FUNCTION: CYCMATCH Mask */

#define DWT_FUNCTION_EMITRANGE_Pos      5                                         /*!< DWT FUNCTION: EMITRANGE Position */

#define DWT_FUNCTION_EMITRANGE_Msk     (0x1UL << DWT_FUNCTION_EMITRANGE_Pos)      /*!< DWT FUNCTION: EMITRANGE Mask */

#define DWT_FUNCTION_FUNCTION_Pos       0                                         /*!< DWT FUNCTION: FUNCTION Position */

#define DWT_FUNCTION_FUNCTION_Msk      (0xFUL /*<< DWT_FUNCTION_FUNCTION_Pos*/) /*!< DWT FUNCTION: FUNCTION Mask */

/*}@*/ /* end of group CMSIS_DWT */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_TPI Trace Port Interface (TPI)
\brief Type definitions for the Trace Port Interface (TPI)
@{

```

```

/** \brief Structure type to access the Trace Port Interface Register (TPI).
*/

```

```

typedef struct
{

```

```

__IO uint32_t SSPSR;           /*!< Offset: 0x000 (R/ ) Supported Parallel Port Size Register */
__IO uint32_t CSPSR;           /*!< Offset: 0x004 (R/W) Current Parallel Port Size Register */

    uint32_t RESERVED0[2];

__IO uint32_t ACPR;            /*!< Offset: 0x010 (R/W) Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];

__IO uint32_t SPPR;            /*!< Offset: 0x0F0 (R/W) Selected Pin Protocol Register */
    uint32_t RESERVED2[131];

__I uint32_t FFSR;             /*!< Offset: 0x300 (R/ ) Formatter and Flush Status Register */
__IO uint32_t FFCR;             /*!< Offset: 0x304 (R/W) Formatter and Flush Control Register */
__I uint32_t FSCR;             /*!< Offset: 0x308 (R/ ) Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];

__I uint32_t TRIGGER;          /*!< Offset: 0xEE8 (R/ ) TRIGGER */
__I uint32_t FIFO0;             /*!< Offset: 0xEEC (R/ ) Integration ETM Data */
__I uint32_t ITATBCTR2;         /*!< Offset: 0xEFO (R/ ) ITATBCTR2 */
    uint32_t RESERVED4[1];

__I uint32_t ITATBCTR0;         /*!< Offset: 0xEF8 (R/ ) ITATBCTR0 */
__I uint32_t FIFO1;             /*!< Offset: 0xEFC (R/ ) Integration ITM Data */
__IO uint32_t ITCTRL;            /*!< Offset: 0xF00 (R/W) Integration Mode Control */
    uint32_t RESERVED5[39];

__IO uint32_t CLAIMSET;          /*!< Offset: 0xFA0 (R/W) Claim tag set */
__IO uint32_t CLAIMCLR;          /*!< Offset: 0xFA4 (R/W) Claim tag clear */
    uint32_t RESERVED7[8];

__I uint32_t DEVID;             /*!< Offset: 0xFC8 (R/ ) TPIU_DEVID */
__I uint32_t DEVTYPE;            /*!< Offset: 0xFCC (R/ ) TPIU_DEVTYPE */

} TPI_Type;

```

/* TPI Asynchronous Clock Prescaler Register Definitions */

```
#define TPI_ACPR_PRESCALER_Pos      0                      /*!< TPI ACPR: PRESCALER Position */
*/
```

```

#define TPI_ACPR_PRESCALER_Msk      (0x1FFFUL /*<< TPI_ACPR_PRESCALER_Pos*/) /*!< TPI
ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */

#define TPI_SPPR_TXMODE_Pos        0                      /*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk        (0x3UL /*<< TPI_SPPR_TXMODE_Pos*/) /*!< TPI SPPR:
TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */

#define TPI_FFSR_FtNonStop_Pos    3                      /*!< TPI FFSR: FtNonStop Position */
#define TPI_FFSR_FtNonStop_Msk    (0x1UL << TPI_FFSR_FtNonStop_Pos) /*!< TPI FFSR:
FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos    2                      /*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk    (0x1UL << TPI_FFSR_TCPresent_Pos) /*!< TPI FFSR:
TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos     1                      /*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk    (0x1UL << TPI_FFSR_FtStopped_Pos) /*!< TPI FFSR:
FtStopped Mask */

#define TPI_FFSR_FInProg_Pos       0                      /*!< TPI FFSR: FInProg Position */
#define TPI_FFSR_FInProg_Msk      (0x1UL /*<< TPI_FFSR_FInProg_Pos*/) /*!< TPI FFSR:
FInProg Mask */

/* TPI Formatter and Flush Control Register Definitions */

#define TPI_FFCR_TrigIn_Pos       8                      /*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk       (0x1UL << TPI_FFCR_TrigIn_Pos) /*!< TPI FFCR: TrigIn Mask
*/
#define TPI_FFCR_EnFCont_Pos      1                      /*!< TPI FFCR: EnFCont Position */

```

```

#define TPI_FFCR_EnFCont_Msk      (0x1UL << TPI_FFCR_EnFCont_Pos)      /*!< TPI FFCR:
EnFCont Mask */

/* TPI TRIGGER Register Definitions */

#define TPI_TRIGGER_TRIGGER_Pos    0                                /*!< TPI TRIGGER: TRIGGER Position */
*/

#define TPI_TRIGGER_TRIGGER_Msk    (0x1UL /*<< TPI_TRIGGER_TRIGGER_Pos*/) /*!< TPI
TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */

#define TPI_FIFO0_ITM_ATVALID_Pos  29                             /*!< TPI FIFO0: ITM_ATVALID
Position */

#define TPI_FIFO0_ITM_ATVALID_Msk  (0x3UL << TPI_FIFO0_ITM_ATVALID_Pos) /*!< TPI FIFO0:
ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos 27                            /*!< TPI FIFO0: ITM_bytecount
Position */

#define TPI_FIFO0_ITM_bytecount_Msk (0x3UL << TPI_FIFO0_ITM_bytecount_Pos) /*!< TPI
FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos  26                            /*!< TPI FIFO0: ETM_ATVALID
Position */

#define TPI_FIFO0_ETM_ATVALID_Msk  (0x3UL << TPI_FIFO0_ETM_ATVALID_Pos) /*!< TPI
FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos 24                            /*!< TPI FIFO0: ETM_bytecount
Position */

#define TPI_FIFO0_ETM_bytecount_Msk (0x3UL << TPI_FIFO0_ETM_bytecount_Pos) /*!< TPI
FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos         16                            /*!< TPI FIFO0: ETM2 Position */

#define TPI_FIFO0_ETM2_Msk        (0xFFUL << TPI_FIFO0_ETM2_Pos) /*!< TPI FIFO0: ETM2
Mask */

```

```

#define TPI_FIFO0_ETM1_Pos      8           /*!< TPI FIFO0: ETM1 Position */
#define TPI_FIFO0_ETM1_Msk      (0xFFUL << TPI_FIFO0_ETM1_Pos)    /*!< TPI FIFO0: ETM1
Mask */

#define TPI_FIFO0_ETM0_Pos      0           /*!< TPI FIFO0: ETM0 Position */
#define TPI_FIFO0_ETM0_Msk      (0xFFUL /*<< TPI_FIFO0_ETM0_Pos*/)  /*!< TPI FIFO0:
ETM0 Mask */

/* TPI ITATBCTR2 Register Definitions */

#define TPI_ITATBCTR2_ATREADY_Pos 0           /*!< TPI ITATBCTR2: ATREADY
Position */
#define TPI_ITATBCTR2_ATREADY_Msk (0x1UL /*<< TPI_ITATBCTR2_ATREADY_Pos*/) /*!< TPI
ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */

#define TPI_FIFO1_ITM_ATVALID_Pos 29          /*!< TPI FIFO1: ITM_ATVALID
Position */
#define TPI_FIFO1_ITM_ATVALID_Msk (0x3UL << TPI_FIFO1_ITM_ATVALID_Pos) /*!< TPI FIFO1:
ITM_ATVALID Mask */

#define TPI_FIFO1_ITM_bytecount_Pos 27         /*!< TPI FIFO1: ITM_bytecount
Position */
#define TPI_FIFO1_ITM_bytecount_Msk (0x3UL << TPI_FIFO1_ITM_bytecount_Pos) /*!< TPI
FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos 26          /*!< TPI FIFO1: ETM_ATVALID
Position */
#define TPI_FIFO1_ETM_ATVALID_Msk (0x3UL << TPI_FIFO1_ETM_ATVALID_Pos) /*!< TPI
FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos 24         /*!< TPI FIFO1: ETM_bytecount
Position */

```

```

#define TPI_FIFO1_ETM_bytecount_Msk      (0x3UL << TPI_FIFO1_ETM_bytecount_Pos) /*!< TPI
FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos              16                      /*!< TPI FIFO1: ITM2 Position */
#define TPI_FIFO1_ITM2_Msk             (0xFFUL << TPI_FIFO1_ITM2_Pos) /*!< TPI FIFO1: ITM2
Mask */

#define TPI_FIFO1_ITM1_Pos              8                       /*!< TPI FIFO1: ITM1 Position */
#define TPI_FIFO1_ITM1_Msk             (0xFFUL << TPI_FIFO1_ITM1_Pos) /*!< TPI FIFO1: ITM1
Mask */

#define TPI_FIFO1_ITM0_Pos              0                       /*!< TPI FIFO1: ITM0 Position */
#define TPI_FIFO1_ITM0_Msk             (0xFFUL /*<< TPI_FIFO1_ITM0_Pos*/) /*!< TPI FIFO1: ITM0
Mask */

/* TPI ITATBCTR0 Register Definitions */

#define TPI_ITATBCTR0_ATREADY_Pos       0                      /*!< TPI ITATBCTR0: ATREADY
Position */
#define TPI_ITATBCTR0_ATREADY_Msk     (0x1UL /*<< TPI_ITATBCTR0_ATREADY_Pos*/) /*!< TPI
ITATBCTR0: ATREADY Mask */

/* TPI Integration Mode Control Register Definitions */

#define TPI_ITCTRL_Mode_Pos            0                      /*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk           (0x1UL /*<< TPI_ITCTRL_Mode_Pos*/) /*!< TPI ITCTRL:
Mode Mask */

/* TPI DEVID Register Definitions */

#define TPI_DEVID_NRZVALID_Pos         11                     /*!< TPI DEVID: NRZVALID Position
*/
#define TPI_DEVID_NRZVALID_Msk        (0x1UL << TPI_DEVID_NRZVALID_Pos) /*!< TPI DEVID:
NRZVALID Mask */

```

```

#define TPI_DEVID_MANCVALID_Pos      10          /*!< TPI DEVID: MANCVALID
Position */

#define TPI_DEVID_MANCVALID_Msk     (0x1UL << TPI_DEVID_MANCVALID_Pos) /*!< TPI
DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALIDID_Pos    9           /*!< TPI DEVID: PTINVALIDID Position
*/

#define TPI_DEVID_PTINVALIDID_Msk   (0x1UL << TPI_DEVID_PTINVALIDID_Pos) /*!< TPI DEVID:
PTINVALIDID Mask */

#define TPI_DEVID_MinBufSz_Pos       6           /*!< TPI DEVID: MinBufSz Position */

#define TPI_DEVID_MinBufSz_Msk      (0x7UL << TPI_DEVID_MinBufSz_Pos) /*!< TPI DEVID:
MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos      5           /*!< TPI DEVID: AsynClkIn Position */

#define TPI_DEVID_AsynClkIn_Msk     (0x1UL << TPI_DEVID_AsynClkIn_Pos) /*!< TPI DEVID:
AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos    0           /*!< TPI DEVID: NrTraceInput
Position */

#define TPI_DEVID_NrTraceInput_Msk   (0xFUL /*<< TPI_DEVID_NrTraceInput_Pos*/) /*!< TPI
DEVID: NrTraceInput Mask */

/* TPI DEVTYPE Register Definitions */

#define TPI_DEVTYPE_MajorType_Pos    4           /*!< TPI DEVTYPE: MajorType
Position */

#define TPI_DEVTYPE_MajorType_Msk   (0xFUL << TPI_DEVTYPE_MajorType_Pos) /*!< TPI
DEVTYPE: MajorType Mask */

#define TPI_DEVTYPE_SubType_Pos      0           /*!< TPI DEVTYPE: SubType Position
*/

#define TPI_DEVTYPE_SubType_Msk     (0xFUL /*<< TPI_DEVTYPE_SubType_Pos*/) /*!< TPI
DEVTYPE: SubType Mask */

```

```

/*@}/* /* end of group CMSIS_TPI */

#ifndef __MPU_PRESENT
#define __MPU_PRESENT 0
#endif

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_MPU Memory Protection Unit (MPU)
 \brief Type definitions for the Memory Protection Unit (MPU)

 @{

 */

/** \brief Structure type to access the Memory Protection Unit (MPU).
 */
typedef struct
{
    __I uint32_t TYPE;           /*!< Offset: 0x000 (R/ ) MPU Type Register          */
    __IO uint32_t CTRL;          /*!< Offset: 0x004 (R/W) MPU Control Register        */
    __IO uint32_t RNR;           /*!< Offset: 0x008 (R/W) MPU Region RNRber Register   */
    __IO uint32_t RBAR;          /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
    __IO uint32_t RASR;          /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register */
    __IO uint32_t RBAR_A1;        /*!< Offset: 0x014 (R/W) MPU Alias 1 Region Base Address Register */
    __IO uint32_t RASR_A1;        /*!< Offset: 0x018 (R/W) MPU Alias 1 Region Attribute and Size Register */
    __IO uint32_t RBAR_A2;        /*!< Offset: 0x01C (R/W) MPU Alias 2 Region Base Address Register */
    __IO uint32_t RASR_A2;        /*!< Offset: 0x020 (R/W) MPU Alias 2 Region Attribute and Size Register */
    __IO uint32_t RBAR_A3;        /*!< Offset: 0x024 (R/W) MPU Alias 3 Region Base Address Register */
} __attribute__((packed));
}
#endif

```

```

__IO uint32_t RASR_A3;           /*!< Offset: 0x028 (R/W) MPU Alias 3 Region Attribute and Size
Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)    /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)    /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)  /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

#define MPU_CTRL_PRIVDEFENA_Pos   2           /*!< MPU CTRL: PRIVDEFENA
Position */
#define MPU_CTRL_PRIVDEFENA_Msk   (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos    1           /*!< MPU CTRL: HFNMIENA
Position */
#define MPU_CTRL_HFNMIENA_Msk    (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos       0           /*!< MPU CTRL: ENABLE Position */

```

```

#define MPU_CTRL_ENABLE_Msk      (1UL /*<< MPU_CTRL_ENABLE_Pos*/)      /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */

#define MPU_RNR_REGION_Pos      0                                /*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk      (0xFFUL /*<< MPU_RNR_REGION_Pos*/)      /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos       5                                /*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk       (0x7FFFFFFUL << MPU_RBAR_ADDR_Pos)      /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos      4                                /*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk      (1UL << MPU_RBAR_VALID_Pos)      /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos     0                                /*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk     (0xFUL /*<< MPU_RBAR_REGION_Pos*/)      /*!< MPU
RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos      16                             /*!< MPU RASR: MPU Region
Attribute field Position */
#define MPU_RASR_ATTRS_Msk      (0xFFFFUL << MPU_RASR_ATTRS_Pos)      /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28                             /*!< MPU RASR: ATTRS.XN Position */
#define MPU_RASR_XN_Msk         (1UL << MPU_RASR_XN_Pos)      /*!< MPU RASR:
ATTRS.XN Mask */

```

#define MPU_RASR_AP_Pos	24	/*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk	(0x7UL << MPU_RASR_AP_Pos)	/*!< MPU RASR: ATTRS.AP Mask */
#define MPU_RASR_TEX_Pos	19	/*!< MPU RASR: ATTRS.TEX Position */
#define MPU_RASR_TEX_Msk	(0x7UL << MPU_RASR_TEX_Pos)	/*!< MPU RASR: ATTRS.TEX Mask */
#define MPU_RASR_S_Pos	18	/*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk	(1UL << MPU_RASR_S_Pos)	/*!< MPU RASR: ATTRS.S Mask */
#define MPU_RASR_C_Pos	17	/*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk	(1UL << MPU_RASR_C_Pos)	/*!< MPU RASR: ATTRS.C Mask */
#define MPU_RASR_B_Pos	16	/*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk	(1UL << MPU_RASR_B_Pos)	/*!< MPU RASR: ATTRS.B Mask */
#define MPU_RASR_SRD_Pos	8	/*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk	(0xFFUL << MPU_RASR_SRD_Pos)	/*!< MPU RASR: Sub-Region Disable Mask */
#define MPU_RASR_SIZE_Pos	1	/*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk	(0x1FUL << MPU_RASR_SIZE_Pos)	/*!< MPU RASR: Region Size Field Mask */

```
#define MPU_RASR_ENABLE_Pos      0           /*!< MPU RASR: Region enable bit  
Position */  
  
#define MPU_RASR_ENABLE_Msk      (1UL /*<< MPU_RASR_ENABLE_Pos*/)      /*!< MPU  
RASR: Region enable bit Disable Mask */
```

```
/*@} end of group CMSIS_MPU */
```

```
#endif
```

```
#if (__FPU_PRESENT == 1)  
/** \ingroup CMSIS_core_register  
\\defgroup CMSIS_FPU Floating Point Unit (FPU)  
\\brief Type definitions for the Floating Point Unit (FPU)  
{@  
*/
```

```
/** \\brief Structure type to access the Floating Point Unit (FPU).
```

```
*/
```

```
typedef struct
```

```
{
```

```
    uint32_t RESERVED0[1];  
  
    __IO uint32_t FPCCR;          /*!< Offset: 0x004 (R/W) Floating-Point Context Control Register  
*/  
  
    __IO uint32_t FPCAR;          /*!< Offset: 0x008 (R/W) Floating-Point Context Address Register  
*/  
  
    __IO uint32_t FPDSCR;         /*!< Offset: 0x00C (R/W) Floating-Point Default Status Control  
Register */  
  
    __I  uint32_t MVFR0;          /*!< Offset: 0x010 (R/ ) Media and FP Feature Register 0 */  
    __I  uint32_t MVFR1;          /*!< Offset: 0x014 (R/ ) Media and FP Feature Register 1 */  
    __I  uint32_t MVFR2;          /*!< Offset: 0x018 (R/ ) Media and FP Feature Register 2 */
```

```

} FPU_Type;

/* Floating-Point Context Control Register */

#define FPU_FPCCR_AS PEN_Pos      31           /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk       (1UL << FPU_FPCCR_AS PEN_Pos)      /*!< FPCCR: ASPEN
bit Mask */

#define FPU_FPCCR_LSPEN_Pos      30           /*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSPEN_Msk       (1UL << FPU_FPCCR_LSPEN_Pos)      /*!< FPCCR: LSPEN
bit Mask */

#define FPU_FPCCR_MONRDY_Pos     8            /*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MONRDY_Msk      (1UL << FPU_FPCCR_MONRDY_Pos)      /*!< FPCCR:
MONRDY bit Mask */

#define FPU_FPCCR_BFRDY_Pos      6            /*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFRDY_Msk       (1UL << FPU_FPCCR_BFRDY_Pos)      /*!< FPCCR: BFRDY
bit Mask */

#define FPU_FPCCR_MMRDY_Pos      5            /*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMRDY_Msk       (1UL << FPU_FPCCR_MMRDY_Pos)      /*!< FPCCR:
MMRDY bit Mask */

#define FPU_FPCCR_HFRDY_Pos      4            /*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk       (1UL << FPU_FPCCR_HFRDY_Pos)      /*!< FPCCR:
HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos      3           /*!< FPCCR: processor mode bit
Position */
#define FPU_FPCCR_THREAD_Msk      (1UL << FPU_FPCCR_THREAD_Pos)      /*!< FPCCR:
processor mode active bit Mask */

```

```

#define FPU_FPCCR_USER_Pos           1           /*!< FPCCR: privilege level bit
Position */

#define FPU_FPCCR_USER_Msk          (1UL << FPU_FPCCR_USER_Pos)      /*!< FPCCR:
privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos        0           /*!< FPCCR: Lazy state preservation
active bit Position */

#define FPU_FPCCR_LSPACT_Msk        (1UL /*<< FPU_FPCCR_LSPACT_Pos*/)    /*!< FPCCR: Lazy
state preservation active bit Mask */

/* Floating-Point Context Address Register */

#define FPU_FPCAR_ADDRESS_Pos       3           /*!< FPCAR: ADDRESS bit Position
*/
#define FPU_FPCAR_ADDRESS_Msk       (0x1FFFFFFFUL << FPU_FPCAR_ADDRESS_Pos) /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */

#define FPU_FPDSCR_AHP_Pos         26          /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk          (1UL << FPU_FPDSCR_AHP_Pos)      /*!< FPDSCR: AHP
bit Mask */

#define FPU_FPDSCR_DN_Pos          25          /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk          (1UL << FPU_FPDSCR_DN_Pos)      /*!< FPDSCR: DN bit
Mask */

#define FPU_FPDSCR_FZ_Pos          24          /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk          (1UL << FPU_FPDSCR_FZ_Pos)      /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos        22          /*!< FPDSCR: RMode bit Position
*/
#define FPU_FPDSCR_RMode_Msk        (3UL << FPU_FPDSCR_RMode_Pos)      /*!< FPDSCR:
RMode bit Mask */

```

```

/* Media and FP Feature Register 0 */

#define FPU_MVFR0_FP_rounding_modes_Pos 28          /*!< MVFR0: FP rounding
modes bits Position */

#define FPU_MVFR0_FP_rounding_modes_Msk (0xFUL << FPU_MVFR0_FP_rounding_modes_Pos)
/*!< MVFR0: FP rounding modes bits Mask */

#define FPU_MVFR0_Short_vectors_Pos    24          /*!< MVFR0: Short vectors bits
Position */

#define FPU_MVFR0_Short_vectors_Msk   (0xFUL << FPU_MVFR0_Short_vectors_Pos)  /*!<
MVFR0: Short vectors bits Mask */

#define FPU_MVFR0_Square_root_Pos     20          /*!< MVFR0: Square root bits
Position */

#define FPU_MVFR0_Square_root_Msk    (0xFUL << FPU_MVFR0_Square_root_Pos)  /*!<
MVFR0: Square root bits Mask */

#define FPU_MVFR0_Divide_Pos         16          /*!< MVFR0: Divide bits Position */

#define FPU_MVFR0_Divide_Msk        (0xFUL << FPU_MVFR0_Divide_Pos)  /*!< MVFR0:
Divide bits Mask */

#define FPU_MVFR0_FP_excep_trapping_Pos 12          /*!< MVFR0: FP exception
trapping bits Position */

#define FPU_MVFR0_FP_excep_trapping_Msk (0xFUL << FPU_MVFR0_FP_excep_trapping_Pos) /*!<
MVFR0: FP exception trapping bits Mask */

#define FPU_MVFR0_Double_precision_Pos 8           /*!< MVFR0: Double-precision
bits Position */

#define FPU_MVFR0_Double_precision_Msk (0xFUL << FPU_MVFR0_Double_precision_Pos) /*!<
MVFR0: Double-precision bits Mask */

#define FPU_MVFR0_Single_precision_Pos 4           /*!< MVFR0: Single-precision
bits Position */

```

```

#define FPU_MVFR0_Single_precision_Msk  (0xFUL << FPU_MVFR0_Single_precision_Pos) /*!<
MVFR0: Single-precision bits Mask */

#define FPU_MVFR0_A SIMD_registers_Pos    0                                /*!< MVFR0: A SIMD registers
bits Position */

#define FPU_MVFR0_A SIMD_registers_Msk  (0xFUL /*<< FPU_MVFR0_A SIMD_registers_Pos*/) /*!<
MVFR0: A SIMD registers bits Mask */

/* Media and FP Feature Register 1 */

#define FPU_MVFR1_FP_fused_MAC_Pos     28                               /*!< MVFR1: FP fused MAC
bits Position */

#define FPU_MVFR1_FP_fused_MAC_Msk   (0xFUL << FPU_MVFR1_FP_fused_MAC_Pos) /*!<
MVFR1: FP fused MAC bits Mask */

#define FPU_MVFR1_FP_HPFP_Pos        24                               /*!< MVFR1: FP HPFP bits
Position */

#define FPU_MVFR1_FP_HPFP_Msk      (0xFUL << FPU_MVFR1_FP_HPFP_Pos) /*!< MVFR1:
FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos     4                                /*!< MVFR1: D_NaN mode bits
Position */

#define FPU_MVFR1_D_NaN_mode_Msk   (0xFUL << FPU_MVFR1_D_NaN_mode_Pos) /*!<
MVFR1: D_NaN mode bits Mask */

#define FPU_MVFR1_FtZ_mode_Pos       0                               /*!< MVFR1: FtZ mode bits
Position */

#define FPU_MVFR1_FtZ_mode_Msk     (0xFUL /*<< FPU_MVFR1_FtZ_mode_Pos*/) /*!<
MVFR1: FtZ mode bits Mask */

/* Media and FP Feature Register 2 */

/*@} end of group CMSIS_FPU */
#endif

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CoreDebug    Core Debug Registers (CoreDebug)
 \brief    Type definitions for the Core Debug Registers

@{

*/
/* ** \brief Structure type to access the Core Debug Register (CoreDebug).
 */
typedef struct
{
    __IO uint32_t DHCSR;           /*!< Offset: 0x000 (R/W) Debug Halting Control and Status Register */
    __O  uint32_t DCRSR;          /*!< Offset: 0x004 ( /W) Debug Core Register Selector Register      */
    __IO uint32_t DCRDR;          /*!< Offset: 0x008 (R/W) Debug Core Register Data Register        */
    __IO uint32_t DEMCR;          /*!< Offset: 0x00C (R/W) Debug Exception and Monitor Control
Register */

} CoreDebug_Type;

/* Debug Halting Control and Status Register */

#define CoreDebug_DHCSR_DBGKEY_Pos     16           /*!< CoreDebug DHCSR:
DBGKEY Position */

#define CoreDebug_DHCSR_DBGKEY_Msk     (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos) /*!<
CoreDebug DHCSR: DBGKEY Mask */

#define CoreDebug_DHCSR_S_RESET_ST_Pos  25           /*!< CoreDebug DHCSR:
S_RESET_ST Position */

#define CoreDebug_DHCSR_S_RESET_ST_Msk (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos) /*!<
CoreDebug DHCSR: S_RESET_ST Mask */

```

```
#define CoreDebug_DHCSR_S_RETIRE_ST_Pos 24          /*!< CoreDebug DHCSR:  
S_RETIRE_ST Position */  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Msk  (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RETIRE_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_LOCKUP_Pos    19          /*!< CoreDebug DHCSR:  
S_LOCKUP Position */  
  
#define CoreDebug_DHCSR_S_LOCKUP_Msk   (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)  /*!<  
CoreDebug DHCSR: S_LOCKUP Mask */  
  
  
#define CoreDebug_DHCSR_S_SLEEP_Pos    18          /*!< CoreDebug DHCSR:  
S_SLEEP Position */  
  
#define CoreDebug_DHCSR_S_SLEEP_Msk   (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)  /*!<  
CoreDebug DHCSR: S_SLEEP Mask */  
  
  
#define CoreDebug_DHCSR_S_HALT_Pos     17          /*!< CoreDebug DHCSR:  
S_HALT Position */  
  
#define CoreDebug_DHCSR_S_HALT_Msk   (1UL << CoreDebug_DHCSR_S_HALT_Pos)  /*!<  
CoreDebug DHCSR: S_HALT Mask */  
  
  
#define CoreDebug_DHCSR_S_REGRDY_Pos   16          /*!< CoreDebug DHCSR:  
S_REGRDY Position */  
  
#define CoreDebug_DHCSR_S_REGRDY_Msk  (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)  /*!<  
CoreDebug DHCSR: S_REGRDY Mask */  
  
  
#define CoreDebug_DHCSR_C_SNAPSTALL_Pos 5           /*!< CoreDebug DHCSR:  
C_SNAPSTALL Position */  
  
#define CoreDebug_DHCSR_C_SNAPSTALL_Msk (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)  
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */  
  
  
#define CoreDebug_DHCSR_C_MASKINTS_Pos 3            /*!< CoreDebug DHCSR:  
C_MASKINTS Position */  
  
#define CoreDebug_DHCSR_C_MASKINTS_Msk (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)  
/*!< CoreDebug DHCSR: C_MASKINTS Mask */
```

```

#define CoreDebug_DHCSR_C_STEP_Pos      2                  /*!< CoreDebug DHCSR: C_STEP
Position */

#define CoreDebug_DHCSR_C_STEP_Msk      (1UL << CoreDebug_DHCSR_C_STEP_Pos)      /*!<
CoreDebug DHCSR: C_STEP Mask */

#define CoreDebug_DHCSR_C_HALT_Pos      1                  /*!< CoreDebug DHCSR:
C_HALT Position */

#define CoreDebug_DHCSR_C_HALT_Msk      (1UL << CoreDebug_DHCSR_C_HALT_Pos)      /*!<
CoreDebug DHCSR: C_HALT Mask */

#define CoreDebug_DHCSR_C_DEBUGEN_Pos    0                  /*!< CoreDebug DHCSR:
C_DEBUGEN Position */

#define CoreDebug_DHCSR_C_DEBUGEN_Msk    (1UL /*<< CoreDebug_DHCSR_C_DEBUGEN_Pos*/) /*!<
CoreDebug DHCSR: C_DEBUGEN Mask */

/* Debug Core Register Selector Register */

#define CoreDebug_DCRSR_REGWnR_Pos      16                 /*!< CoreDebug DCRSR:
REGWnR Position */

#define CoreDebug_DCRSR_REGWnR_Msk      (1UL << CoreDebug_DCRSR_REGWnR_Pos)      /*!<
CoreDebug DCRSR: REGWnR Mask */

#define CoreDebug_DCRSR_REGSEL_Pos      0                  /*!< CoreDebug DCRSR: REGSEL
Position */

#define CoreDebug_DCRSR_REGSEL_Msk      (0x1FUL /*<< CoreDebug_DCRSR_REGSEL_Pos*/) /*!<
CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos      24                 /*!< CoreDebug DEMCR:
TRCENA Position */

#define CoreDebug_DEMCR_TRCENA_Msk      (1UL << CoreDebug_DEMCR_TRCENA_Pos)      /*!<
CoreDebug DEMCR: TRCENA Mask */

```

```
#define CoreDebug_DEMCR_MON_REQ_Pos    19           /*!< CoreDebug DEMCR:  
MON_REQ Position */  
  
#define CoreDebug_DEMCR_MON_REQ_Msk   (1UL << CoreDebug_DEMCR_MON_REQ_Pos)  
/*!< CoreDebug DEMCR: MON_REQ Mask */  
  
  
#define CoreDebug_DEMCR_MON_STEP_Pos   18           /*!< CoreDebug DEMCR:  
MON_STEP Position */  
  
#define CoreDebug_DEMCR_MON_STEP_Msk  (1UL << CoreDebug_DEMCR_MON_STEP_Pos)  
/*!< CoreDebug DEMCR: MON_STEP Mask */  
  
  
#define CoreDebug_DEMCR_MON_PEND_Pos   17           /*!< CoreDebug DEMCR:  
MON_PEND Position */  
  
#define CoreDebug_DEMCR_MON_PEND_Msk  (1UL << CoreDebug_DEMCR_MON_PEND_Pos)  
/*!< CoreDebug DEMCR: MON_PEND Mask */  
  
  
#define CoreDebug_DEMCR_MON_EN_Pos    16           /*!< CoreDebug DEMCR:  
MON_EN Position */  
  
#define CoreDebug_DEMCR_MON_EN_Msk   (1UL << CoreDebug_DEMCR_MON_EN_Pos)      /*!<  
CoreDebug DEMCR: MON_EN Mask */  
  
  
#define CoreDebug_DEMCR_VC_HARDERR_Pos 10           /*!< CoreDebug DEMCR:  
VC_HARDERR Position */  
  
#define CoreDebug_DEMCR_VC_HARDERR_Msk (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos)  
/*!< CoreDebug DEMCR: VC_HARDERR Mask */  
  
  
#define CoreDebug_DEMCR_VC_INTERR_Pos  9            /*!< CoreDebug DEMCR:  
VC_INTERR Position */  
  
#define CoreDebug_DEMCR_VC_INTERR_Msk (1UL << CoreDebug_DEMCR_VC_INTERR_Pos)     /*!<  
CoreDebug DEMCR: VC_INTERR Mask */  
  
  
#define CoreDebug_DEMCR_VC_BUSERR_Pos  8            /*!< CoreDebug DEMCR:  
VC_BUSERR Position */  
  
#define CoreDebug_DEMCR_VC_BUSERR_Msk (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos)  
/*!< CoreDebug DEMCR: VC_BUSERR Mask */
```

```

#define CoreDebug_DEMCR_VC_STATERR_Pos    7           /*!< CoreDebug DEMCR:  

VC_STATERR Position */  
  

#define CoreDebug_DEMCR_VC_STATERR_Msk   (1UL << CoreDebug_DEMCR_VC_STATERR_Pos)  

/*!< CoreDebug DEMCR: VC_STATERR Mask */  
  

#define CoreDebug_DEMCR_VC_CHKERR_Pos    6           /*!< CoreDebug DEMCR:  

VC_CHKERR Position */  
  

#define CoreDebug_DEMCR_VC_CHKERR_Msk   (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos)  

/*!< CoreDebug DEMCR: VC_CHKERR Mask */  
  

#define CoreDebug_DEMCR_VC_NOCPERR_Pos   5           /*!< CoreDebug DEMCR:  

VC_NOCPERR Position */  
  

#define CoreDebug_DEMCR_VC_NOCPERR_Msk  (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos)  

/*!< CoreDebug DEMCR: VC_NOCPERR Mask */  
  

#define CoreDebug_DEMCR_VC_MMERR_Pos    4           /*!< CoreDebug DEMCR:  

VC_MMERR Position */  
  

#define CoreDebug_DEMCR_VC_MMERR_Msk   (1UL << CoreDebug_DEMCR_VC_MMERR_Pos)  

/*!< CoreDebug DEMCR: VC_MMERR Mask */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Pos 0           /*!< CoreDebug DEMCR:  

VC_CORERESET Position */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Msk (1UL /*<<  

CoreDebug_DEMCR_VC_CORERESET_Pos*/) /*!< CoreDebug DEMCR: VC_CORERESET Mask */

```

/*@} end of group CMSIS_CoreDebug */

```

/** \ingroup CMSIS_core_register  

\defgroup CMSIS_core_base Core Definitions  

\brief Definitions for base addresses, unions, and structures.

```

```

@{

/* Memory mapping of Cortex-M4 Hardware */

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */
#define ITM_BASE      (0xE0000000UL)           /*!< ITM Base Address */
#define DWT_BASE      (0xE0001000UL)           /*!< DWT Base Address */
#define TPI_BASE      (0xE0040000UL)           /*!< TPI Base Address */
#define CoreDebug_BASE (0xE000EDFOUL)          /*!< Core Debug Base Address */
#define SysTick_BASE   (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */
#define NVIC_BASE     (SCS_BASE + 0x0100UL)          /*!< NVIC Base Address */
#define SCB_BASE       (SCS_BASE + 0x0D00UL)          /*!< System Control Block Base Address */

#define SCnSCB        ((SCnSCB_Type *) SCS_BASE)    /*!< System control Register not in SCB */
#define SCB           ((SCB_Type *) SCB_BASE)         /*!< SCB configuration struct */
#define SysTick       ((SysTick_Type *) SysTick_BASE)  /*!< SysTick configuration struct */
#define NVIC          ((NVIC_Type *) NVIC_BASE)        /*!< NVIC configuration struct */
#define ITM           ((ITM_Type *) ITM_BASE)          /*!< ITM configuration struct */
#define DWT           ((DWT_Type *) DWT_BASE)          /*!< DWT configuration struct */
#define TPI           ((TPI_Type *) TPI_BASE)          /*!< TPI configuration struct */
#define CoreDebug     ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration
struct */

#if (__MPU_PRESENT == 1)

#define MPU_BASE      (SCS_BASE + 0x0D90UL)          /*!< Memory Protection Unit */
#define MPU          ((MPU_Type *) MPU_BASE)          /*!< Memory Protection Unit */

#endif

#if (__FPU_PRESENT == 1)

```

```
#define FPU_BASE      (SCS_BASE + 0x0F30UL)           /*!< Floating Point Unit      */
#define FPU      ((FPU_Type    *)  FPU_BASE   ) /*!< Floating Point Unit      */
#endif

/*@} */
```

```
*****
```

```
*       Hardware Abstraction Layer
```

Core Function Interface contains:

- Core NVIC Functions
- Core SysTick Functions
- Core Debug Functions
- Core Register Access Functions

```
*****/
```

```
/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/
```

```
/* ##### NVIC functions ##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the NVIC.
@{
*/
```

```
/** \brief Set Priority Grouping
```

The function sets the priority grouping field using the required unlock sequence.

The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.

Only values from 0..7 are used.

In case of a conflict between priority grouping and available

priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

```
\param [in] PriorityGroup Priority grouping field.  
*/  
__STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)  
{  
    uint32_t reg_value;  
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL);      /* only values 0..7 are used  
 */  
  
    reg_value = SCB->AIRCR;                      /* read old register configuration */  
    reg_value &= ~((uint32_t)(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk));      /* clear  
 bits to change */  
    reg_value = (reg_value  
                |  
                ((uint32_t)0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |  
                (PriorityGroupTmp << 8));      /* Insert write key and priority group */  
    SCB->AIRCR = reg_value;  
}  
  
/** \brief Get Priority Grouping
```

The function reads the priority grouping field from the NVIC Interrupt Controller.

```
\return Priority grouping field (SCB->AIRCR [10:8] PRIGROUP field).  
*/  
STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)  
{  
    return ((uint32_t)((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos));  
}
```

```
/** \brief Enable External Interrupt
```

The function enables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
0x1FUL));  
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)  
{
```

```
NVIC->ICER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
0x1FUL));  
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)  
{  
    return((uint32_t)((NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn)  
& 0x1FUL))) != 0UL) ? 1UL : 0UL);  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

\param [in] IRQn Interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
```

```
{  
    NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
    0x1FUL));  
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```
\param [in]    IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
    0x1FUL));  
}
```

```
/** \brief Get Active Interrupt
```

The function reads the active register in NVIC and returns the active bit.

```
\param [in]    IRQn Interrupt number.  
  
\return      0 Interrupt status is not active.  
\return      1 Interrupt status is active.  
*/  
STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
```

```
{  
    return((uint32_t)((NVIC->IABR[((uint32_t)(int32_t)IRQn) >> 5UL] & (1UL << (((uint32_t)(int32_t)IRQn)  
    & 0x1FUL)) != 0UL) ? 1UL : 0UL));  
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)  
{  
    if((int32_t)IRQn < 0) {  
        SCB->SHPR[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS))  
        & (uint32_t)0xFFUL);  
    }  
    else {  
        NVIC->IP[((uint32_t)(int32_t)IRQn)] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &  
        (uint32_t)0xFFUL);  
    }  
}
```

```
/** \brief Get Interrupt Priority
```

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

```
\param [in] IRQn Interrupt number.  
\return Interrupt Priority. Value is aligned automatically to the implemented  
priority bits of the microcontroller.  
*/  
STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)  
{  
  
if((int32_t)IRQn < 0) {  
    return(((uint32_t)SCB->SHPR[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] >> (8 - __NVIC_PRIO_BITS));  
}  
else {  
    return(((uint32_t)NVIC->IP[((uint32_t)(int32_t)IRQn)] >> (8 - __NVIC_PRIO_BITS));  
}  
}  
  
/** \brief Encode Priority
```

The function encodes the priority for an interrupt with the given priority group, preemptive priority value, and subpriority value.

In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS), the smallest possible priority group is set.

```
\param [in] PriorityGroup Used priority group.
```

```

\param [in] PreemptPriority Preemptive priority value (starting from 0).
\param [in] SubPriority Subpriority value (starting from 0).
\return Encoded priority. Value can be used in the function \ref NVIC_SetPriority().

*/
__STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority,
                                            uint32_t SubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
        (uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);

    SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
        (uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));

    return (
        ((PreemptPriority & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL)) << SubPriorityBits) |
        ((SubPriority & (uint32_t)((1UL << (SubPriorityBits)) - 1UL)))
    );
}
/** \brief Decode Priority

```

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

```

    \param [in]      Priority  Priority value, which can be retrieved with the function \ref
NVIC_GetPriority().

    \param [in]  PriorityGroup  Used priority group.

    \param [out] pPreemptPriority  Preemptive priority value (starting from 0).

    \param [out]  pSubPriority  Subpriority value (starting from 0).

*/
__STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t*
pPreemptPriority, uint32_t* pSubPriority)

{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
(uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);

    SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
(uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));

    *pPreemptPriority = (Priority >> SubPriorityBits) & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL);
    *pSubPriority = (Priority ) & (uint32_t)((1UL << (SubPriorityBits )) - 1UL);
}

/** \brief System Reset

```

The function initiates a system reset request to reset the MCU.

```

*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB(); /* Ensure all outstanding memory accesses included

```

```

        buffered write are completed before reset */

SCB->AIRCR = (uint32_t)((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
    (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
    SCB_AIRCR_SYSRESETREQ_Msk ); /* Keep priority group unchanged */

__DSB(); /* Ensure completion of memory access */

while(1) { __NOP(); } /* wait until reset */

}

```

/*@} end of CMSIS_Core_NVICFunctions */

/* ##### FPU functions ##### */

```

/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_FpuFunctions FPU Functions
\brief Function that provides FPU type.

@{
*/

```

/**

\fn uint32_t SCB_GetFPType(void)

\brief get FPU type

\returns

- \b 0: No FPU

- \b 1: Single precision FPU

- \b 2: Double + Single precision FPU

*/

__STATIC_INLINE uint32_t SCB_GetFPType(void)

{

uint32_t mvfr0;

```
mvfr0 = SCB->MVFR0;

if ((mvfr0 & 0x00000FF0UL) == 0x220UL) {
    return 2UL;      // Double + Single precision FPU
} else if ((mvfr0 & 0x00000FF0UL) == 0x020UL) {
    return 1UL;      // Single precision FPU
} else {
    return 0UL;      // No FPU
}
```

```
/*@} end of CMSIS_Core_FpuFunctions */
```

```
/* ##### Cache functions ##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_CacheFunctions Cache Functions
 \brief Functions that configure Instruction and Data cache.

 @{
 */

```

```
/* Cache Size ID Register Macros */
```

```
#define CCSIDR_WAYS(x) (((x) & SCB_CCSIDR_ASSOCIATIVITY_Msk) >>
SCB_CCSIDR_ASSOCIATIVITY_Pos)

#define CCSIDR_SETS(x) (((x) & SCB_CCSIDR_NUMSETS_Msk) >> SCB_CCSIDR_NUMSETS_Pos)

#define CCSIDR_LSSHIFT(x) (((x) & SCB_CCSIDR_LINESIZE_Msk) /*>> SCB_CCSIDR_LINESIZE_Pos*/)
```

```
/** \brief Enable I-Cache

The function turns on I-Cache

*/
__STATIC_INLINE void SCB_EnableICache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->ICIALLU = 0UL;           // invalidate I-Cache
    SCB->CCR |= (uint32_t)SCB_CCR_IC_Msk; // enable I-Cache
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Disable I-Cache

The function turns off I-Cache

*/
__STATIC_INLINE void SCB_DisableICache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->CCR &= ~(uint32_t)SCB_CCR_IC_Msk; // disable I-Cache
    SCB->ICIALLU = 0UL;           // invalidate I-Cache
}
```

```
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Invalidate I-Cache
```

The function invalidates I-Cache

```
 */
__STATIC_INLINE void SCB_InvalidateCache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->ICIALLU = 0UL;
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Enable D-Cache
```

The function turns on D-Cache

```
 */
__STATIC_INLINE void SCB_EnableDCache (void)
{
#if (__DCACHE_PRESENT == 1)
```

```

uint32_t ccsidr, sshift, wshift, sw;

uint32_t sets, ways;

SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
ccsidr = SCB->CCSIDR;
sets = (uint32_t)(CCSIDR_SETS(ccsidr));
sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

__DSB();

do {                                // invalidate D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCISW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();

SCB->CCR |= (uint32_t)SCB_CCR_DC_Msk; // enable D-Cache

__DSB();
__ISB();
#endif
}

```

```

/** \brief Disable D-Cache

The function turns off D-Cache

*/
__STATIC_INLINE void SCB_DisableDCache (void)
{
#if (__DCACHE_PRESENT == 1)

    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

    __DSB();

    SCB->CCR &= ~(uint32_t)SCB_CCR_DC_Msk; // disable D-Cache

    do {                                // clean & invalidate D-Cache
        uint32_t tmpways = ways;
        do {
            sw = ((tmpways << wshift) | (sets << sshift));
            SCB->DCCISW = sw;
        } while(tmpways--);
    } while(sets--);
}

```

```
    __DSB();  
    __ISB();  
#endif  
}  
  
/*\brief Invalidate D-Cache
```

The function invalidates D-Cache

```
 */  
__STATIC_INLINE void SCB_InvalidateDCache (void)  
{  
#if (__DCACHE_PRESENT == 1)  
    uint32_t ccsidr, sshift, wshift, sw;  
    uint32_t sets, ways;  
  
    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache  
    ccsidr = SCB->CCSIDR;  
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));  
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);  
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));  
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);  
  
    __DSB();  
  
    do {  
        // invalidate D-Cache  
        uint32_t tmpways = ways;  
        do {
```

```

    sw = ((tmpways << wshift) | (sets << sshift));
    SCB->DCISW = sw;
} while(tmpways--);

} while(sets--);

__DSB();
__ISB();

#endif
}

/** \brief Clean D-Cache

The function cleans D-Cache

*/
__STATIC_INLINE void SCB_CleanDCache (void)
{
#ifndef __DCACHE_PRESENT == 1
    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (OUL << 1) | OUL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

    __DSB();

```

```

do {                                // clean D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCCSW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();
__ISB();
#endif
}

/** \brief Clean & Invalidate D-Cache

The function cleans and Invalidates D-Cache
*/
__STATIC_INLINE void SCB_CleanInvalidateDCache (void)
{
#if (__DCACHE_PRESENT == 1)
    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);

```

```

ways  = (uint32_t)(CCSIDR_WAYS(ccsidr));
wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

__DSB();

do {           // clean & invalidate D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCCISW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();
__ISB();
#endif
}

/***
\fn     void SCB_InvalidateDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief   D-Cache Invalidate by address
\param[in]  addr  address (aligned to 32-byte boundary)
\param[in]  dsize  size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_InvalidateDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;

```

```

    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;           // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

    __DSB();

    while (op_size > 0) {
        SCB->DCIMVAC = op_addr;
        op_addr +=     linesize;
        op_size -= (int32_t)linesize;
    }

    __DSB();
    __ISB();
#endif
}

/***
\fn      void SCB_CleanDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief   D-Cache Clean by address
\param[in] addr  address (aligned to 32-byte boundary)
\param[in] dsize size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_CleanDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;
    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;           // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

```

```

__DSB();

while (op_size > 0) {
    SCB->DCCMVAC = op_addr;
    op_addr +=     linesize;
    op_size -= (int32_t)linesize;
}

__DSB();
__ISB();
#endif
}

/***
\fn      void SCB_CleanInvalidateDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief    D-Cache Clean and Invalidate by address
\param[in]  addr  address (aligned to 32-byte boundary)
\param[in]  dsize size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_CleanInvalidateDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;
    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;          // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

    __DSB();

```

```
while (op_size > 0) {  
    SCB->DCCIMVAC = op_addr;  
    op_addr +=     linesize;  
    op_size -= (int32_t)linesize;  
}  
  
__DSB();  
__ISB();  
#endif  
}  
  
/*@} end of CMSIS_Core_CacheFunctions */
```

```
/* ##### SysTick function  
##### */  
/** \ingroup CMSIS_Core_FunctionInterface  
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions  
 \brief Functions that configure the System.  
 @  
 */
```

```
#if (__Vendor_SysTickConfig == 0)  
  
/** \brief System Tick Configuration
```

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.

Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

```
\return 0 Function succeeded.
```

```
\return      1 Function failed.
```

\note When the variable **_Vendor_SysTickConfig** is set to 1, then the function **SysTick_Config** is not included. In this case, the file **device.h** must contain a vendor-specific implementation of this function.

* /

```

__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)

{
if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) { return (1UL); } /* Reload value impossible */

SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */

NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
*/
SysTick->VAL = 0UL; /* Load the SysTick Counter Value */

SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
    SysTick_CTRL_TICKINT_Msk |
    SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */

return (0UL); /* Function successful */
}

#endif

```

```

/*@} end of CMSIS_Core_SysTickFunctions */

/*
***** Debug In/Output function
***** */

/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_core_DebugFunctions ITM Functions
 \brief Functions that access the ITM debug interface.

 @{
 */

extern volatile int32_t ITM_RxBuffer;           /*!< External variable to receive characters.
 */

#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is
ready for next character. */

/** \brief ITM Send Character

The function transmits a character via the ITM channel 0, and
\li Just returns when no debugger is connected that has booked the output.
\li Is blocking when a debugger is connected, but the previous character sent has not been
transmitted.

\param [in] ch Character to transmit.

\returns Character to transmit.

*/
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)

```

```

{
if (((ITM->TCR & ITM_TCR_ITMENA_Msk) != 0UL) && /* ITM enabled */
    ((ITM->TER & 1UL ) != 0UL) ) /* ITM Port #0 enabled */

{
    while (ITM->PORT[0].u32 == 0UL) { __NOP(); }

    ITM->PORT[0].u8 = (uint8_t)ch;

}
return (ch);
}

```

/** \brief ITM Receive Character

The function inputs a character via the external variable \ref ITM_RxBuffer.

```

\return Received character.

\return -1 No character pending.

*/
__STATIC_INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1; /* no character available */

    if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
        ch = ITM_RxBuffer;
        ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
    }

    return (ch);
}

```

```
/** \brief ITM Check Character
```

The function checks whether a character is pending for reading in the variable \ref ITM_RxBuffer.

```
\return 0 No character available.  
\return 1 Character available.  
*/  
STATIC_INLINE int32_t ITM_CheckChar (void) {  
  
    if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {  
        return (0); /* no character available */  
    } else {  
        return (1); /* character available */  
    }  
}
```

```
/*@} end of CMSIS_core_DebugFunctions */
```

```
#ifdef __cplusplus  
}  
#endif
```

```
#endif /* __CORE_CM7_H_DEPENDANT */
```

```
#endif /* __CMSIS_GENERIC */
```

```
/*
 * uart.h
 *
 * Created on: Oct 26, 2017
 * Author: modak
 */
```

```
#ifndef INCLUDES_UART_H_
#define INCLUDES_UART_H_

#include <stdio.h>
#include <stdint.h>
#include "MKL25Z4.h"

#if baud==115200
    #define uart_reg->UART0_BDL 0x17
    #define uart_reg->UART0_BDH 0x00
    #define uart_reg->UART0_C4 0x0F
#elif baud==38400
    #define uart_reg->UART0_BDL 0x44
    #define uart_reg->UART0_BDH 0x00
    #define uart_reg->UART0_C4 0x0F
#else
    #define uart_reg->UART0_BDL 0x2E
    #define uart_reg->UART0_BDH 0x00
    #define uart_reg->UART0_C4 0x0F
#endif
```

```
#endif

int UART_configure(void);

int UART_send(char *data_send);

int UART_send_n(char *data_send_n, int length);

int UART_receive(char *data_receive);

int UART_receive_n(char *data_receive_n,int length);

int UART0_IRQHandler(void);

#endif /* INCLUDES_UART_H_ */

/*
** #####
** Processors:      MKL25Z128FM4
**
**          MKL25Z128FT4
**
**          MKL25Z128LH4
**
**          MKL25Z128VLK4
**
** Compilers:      Keil ARM C/C++ Compiler
**
**          Freescale C/C++ for Embedded ARM
**
**          GNU C Compiler
**
**          GNU C Compiler - CodeSourcery Sourcery G++
**
**          IAR ANSI C/C++ Compiler for ARM
**
**
```

** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012

** Version: rev. 2.5, 2015-02-19

** Build: b150220

**

** Abstract:

** CMSIS Peripheral Access Layer for MKL25Z4

**

** Copyright (c) 1997 - 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com
** mail: support@freescale.com

**

** Revisions:

** - rev. 1.0 (2012-06-13)
** Initial version.
** - rev. 1.1 (2012-06-21)
** Update according to reference manual rev. 1.
** - rev. 1.2 (2012-08-01)
** Device type UARTLP changed to UART0.
** - rev. 1.3 (2012-10-04)
** Update according to reference manual rev. 3.
** - rev. 1.4 (2012-11-22)
** MCG module - bit LOLS in MCG_S register renamed to LOLS0.
** NV registers - bit EZPORT_DIS in NV_FOPT register removed.
** - rev. 1.5 (2013-04-05)
** Changed start of doxygen comment.
** - rev. 2.0 (2013-10-29)
** Register accessor macros added to the memory map.
** Symbols for Processor Expert memory map compatibility added to the memory map.
** Startup file for gcc has been updated according to CMSIS 3.2.
** System initialization updated.
** - rev. 2.1 (2014-07-16)

```
**     Module access macro module_BASES replaced by module_BASE_PTRS.  
**     System initialization and startup updated.  
** - rev. 2.2 (2014-08-22)  
**     System initialization updated - default clock config changed.  
** - rev. 2.3 (2014-08-28)  
**     Update of startup files - possibility to override DefaultISR added.  
** - rev. 2.4 (2014-10-14)  
**     Interrupt INT_LPTimer renamed to INT_LPTMR0.  
** - rev. 2.5 (2015-02-19)  
**     Renamed interrupt vector LLW to LLWU.  
**  
** #####  
*/
```

```
/*!  
 * @file MKL25Z4.h  
 * @version 2.5  
 * @date 2015-02-19  
 * @brief CMSIS Peripheral Access Layer for MKL25Z4  
 *  
 * CMSIS Peripheral Access Layer for MKL25Z4  
*/
```

```
/* -----  
-- MCU activation  
----- */
```

```
/* Prevention from multiple including the same memory map */
```

```
#if !defined(MKL25Z4_H_) /* Check if memory map has not been already included */

#define MKL25Z4_H_
#define MCU_MKL25Z4

/* Check if another memory map has not been also included */

#if (defined(MCU_ACTIVE))

#error MKL25Z4 memory map: There is already included another memory map. Only one memory map
can be included.

#endif /* (defined(MCU_ACTIVE)) */

#define MCU_ACTIVE

#include <stdint.h>

/** Memory map major version (memory maps with equal major version number are
 * compatible) */

#define MCU_MEM_MAP_VERSION 0x0200u

/** Memory map minor version */

#define MCU_MEM_MAP_VERSION_MINOR 0x0005u

/*
-----*
-- Interrupt vector numbers
----- */
/*!
* @addtogroup Interrupt_vector_numbers Interrupt vector numbers
* @{
*/

```

```

/** Interrupt Number Definitions */

#define NUMBER_OF_INT_VECTORS 48           /**< Number of interrupts in the Vector table */

typedef enum IRQn {
    /* Core interrupts */

    NonMaskableInt_IRQn      = -14,      /**< Non Maskable Interrupt */
    HardFault_IRQn            = -13,      /**< Cortex-M0 SV Hard Fault Interrupt */
    SVCall_IRQn               = -5,       /**< Cortex-M0 SV Call Interrupt */
    PendSV_IRQn               = -2,       /**< Cortex-M0 Pend SV Interrupt */
    SysTick_IRQn               = -1,       /**< Cortex-M0 System Tick Interrupt */

    /* Device specific interrupts */

    DMA0_IRQn                 = 0,        /**< DMA channel 0 transfer complete */
    DMA1_IRQn                 = 1,        /**< DMA channel 1 transfer complete */
    DMA2_IRQn                 = 2,        /**< DMA channel 2 transfer complete */
    DMA3_IRQn                 = 3,        /**< DMA channel 3 transfer complete */
    Reserved20_IRQn            = 4,        /**< Reserved interrupt */
    FTFA_IRQn                 = 5,        /**< Command complete and read collision */
    LVD_LVW_IRQn               = 6,        /**< Low-voltage detect, low-voltage warning */
    LLWU_IRQn                  = 7,        /**< Low leakage wakeup Unit */
    I2C0_IRQn                  = 8,        /**< I2C0 interrupt */
    I2C1_IRQn                  = 9,        /**< I2C1 interrupt */
    SPI0_IRQn                  = 10,       /**< SPI0 single interrupt vector for all sources */
    SPI1_IRQn                  = 11,       /**< SPI1 single interrupt vector for all sources */
    UART0_IRQn                 = 12,       /**< UART0 status and error */
    UART1_IRQn                 = 13,       /**< UART1 status and error */
    UART2_IRQn                 = 14,       /**< UART2 status and error */
    ADC0_IRQn                  = 15,       /**< ADC0 interrupt */
    CMP0_IRQn                  = 16,       /**< CMP0 interrupt */
}
```

```

TPM0_IRQn      = 17,      /**< TPM0 single interrupt vector for all sources */
TPM1_IRQn      = 18,      /**< TPM1 single interrupt vector for all sources */
TPM2_IRQn      = 19,      /**< TPM2 single interrupt vector for all sources */
RTC_IRQn       = 20,      /**< RTC alarm */
RTC_Seconds_IRQn = 21,    /**< RTC seconds */
PIT_IRQn       = 22,      /**< PIT interrupt */
Reserved39_IRQn = 23,    /**< Reserved interrupt */
USB0_IRQn      = 24,      /**< USB0 interrupt */
DAC0_IRQn      = 25,      /**< DAC0 interrupt */
TSI0_IRQn      = 26,      /**< TSI0 interrupt */
MCG_IRQn       = 27,      /**< MCG interrupt */
LPTMR0_IRQn   = 28,      /**< LPTMR0 interrupt */
Reserved45_IRQn = 29,    /**< Reserved interrupt */
PORTA_IRQn     = 30,      /**< PORTA Pin detect */
PORTD_IRQn     = 31,      /**< PORTD Pin detect */

} IRQn_Type;

```

```

/*!
* @}
*/
/* /* end of group Interrupt_vector_numbers */

```

```

/*
-----  

-- Cortex M0 Core Configuration  

----- */

```

```

/*!  

* @addtogroup Cortex_Core_Configuration Cortex M0 Core Configuration
* @{

```

```

*/



#define __CM0PLUS_REV      0x0000 /*< Core revision r0p0 */
#define __MPU_PRESENT       0     /*< Defines if an MPU is present or not */
#define __VTOR_PRESENT      1     /*< Defines if an MPU is present or not */
#define __NVIC_PRIO_BITS    2     /*< Number of priority bits implemented in the NVIC */
#define __Vendor_SysTickConfig 0     /*< Vendor specific implementation of SysTickConfig is
defined */

#include "core_cm0plus.h"      /* Core Peripheral Access Layer */
#include "system_MKL25Z4.h"    /* Device specific configuration file */

/*!
* @}
* /* end of group Cortex_Core_Configuration */

/*
----- --
-- Device Peripheral Access Layer
----- */
/*!
* @addtogroup Peripheral_access_layer Device Peripheral Access Layer
* @{
*/
/*


** Start of section using anonymous unions

```

```
*/
```

```
#if defined(__ARMCC_VERSION)
#pragma push
#pragma anon_unions
#elif defined(__CWCC__)
#pragma push
#pragma cpp_extensions on
#elif defined(__GNUC__)
/* anonymous unions are enabled by default */
#elif defined(__IAR_SYSTEMS_ICC__)
#pragma language=extended
#else
#error Not supported compiler type
#endif

/* -----
-- ADC Peripheral Access Layer
----- */
/*!
* @addtogroup ADC_Peripheral_Access_Layer ADC Peripheral Access Layer
* @{
*/
/** ADC - Register Layout Typedef */
typedef struct {
    __IO uint32_t SC1[2];           /**< ADC Status and Control Registers 1, array offset: 0x0, array
step: 0x4 */
}
```

```

__IO uint32_t CFG1;           /**< ADC Configuration Register 1, offset: 0x8 */
__IO uint32_t CFG2;           /**< ADC Configuration Register 2, offset: 0xC */
__I uint32_t R[2];            /**< ADC Data Result Register, array offset: 0x10, array step: 0x4 */
__IO uint32_t CV1;            /**< Compare Value Registers, offset: 0x18 */
__IO uint32_t CV2;            /**< Compare Value Registers, offset: 0x1C */
__IO uint32_t SC2;            /**< Status and Control Register 2, offset: 0x20 */
__IO uint32_t SC3;            /**< Status and Control Register 3, offset: 0x24 */
__IO uint32_t OFS;             /**< ADC Offset Correction Register, offset: 0x28 */
__IO uint32_t PG;              /**< ADC Plus-Side Gain Register, offset: 0x2C */
__IO uint32_t MG;              /**< ADC Minus-Side Gain Register, offset: 0x30 */
__IO uint32_t CLPD;            /**< ADC Plus-Side General Calibration Value Register, offset:
0x34 */
__IO uint32_t CLPS;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x38
*/
__IO uint32_t CLP4;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x3C
*/
__IO uint32_t CLP3;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x40
*/
__IO uint32_t CLP2;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x44
*/
__IO uint32_t CLP1;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x48
*/
__IO uint32_t CLP0;            /**< ADC Plus-Side General Calibration Value Register, offset: 0x4C
*/
uint8_t RESERVED_0[4];
__IO uint32_t CLMD;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x54 */
__IO uint32_t CLMS;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x58 */
__IO uint32_t CLM4;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x5C */
__IO uint32_t CLM3;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x60 */

```

```

    __IO uint32_t CLM2;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x64 */

    __IO uint32_t CLM1;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x68 */

    __IO uint32_t CLM0;           /**< ADC Minus-Side General Calibration Value Register, offset:
0x6C */

} ADC_Type, *ADC_MemMapPtr;

```

```

/*
-----  

-- ADC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros  

* @{
* /
```

```

/* ADC - Register accessors */

#define ADC_SC1_REG(base,index)      ((base)->SC1[index])
#define ADC_SC1_COUNT                2
#define ADC_CFG1_REG(base)           ((base)->CFG1)
#define ADC_CFG2_REG(base)           ((base)->CFG2)
#define ADC_R_REG(base,index)        ((base)->R[index])
#define ADC_R_COUNT                  2
#define ADC.CV1_REG(base)            ((base)->CV1)
#define ADC.CV2_REG(base)            ((base)->CV2)
#define ADC_SC2_REG(base)            ((base)->SC2)
#define ADC_SC3_REG(base)            ((base)->SC3)
#define ADC_OFS_REG(base)             ((base)->OFS)
```

```
#define ADC_PG_REG(base) ((base)->PG)
#define ADC_MG_REG(base) ((base)->MG)
#define ADC_CLPD_REG(base) ((base)->CLPD)
#define ADC_CLPS_REG(base) ((base)->CLPS)
#define ADC_CLP4_REG(base) ((base)->CLP4)
#define ADC_CLP3_REG(base) ((base)->CLP3)
#define ADC_CLP2_REG(base) ((base)->CLP2)
#define ADC_CLP1_REG(base) ((base)->CLP1)
#define ADC_CLP0_REG(base) ((base)->CLP0)
#define ADC_CLMD_REG(base) ((base)->CLMD)
#define ADC_CLMS_REG(base) ((base)->CLMS)
#define ADC_CLM4_REG(base) ((base)->CLM4)
#define ADC_CLM3_REG(base) ((base)->CLM3)
#define ADC_CLM2_REG(base) ((base)->CLM2)
#define ADC_CLM1_REG(base) ((base)->CLM1)
#define ADC_CLM0_REG(base) ((base)->CLM0)
```

```
/*
 * @}
 */
/* /* end of group ADC_Register_Accessor_Macros */
```

```
/*
-----  
-- ADC Register Masks  
----- */
```

```
/*
 * @addtogroup ADC_Register_Masks ADC Register Masks
 * @{
```

*/

```
/* SC1 Bit Fields */

#define ADC_SC1_ADCH_MASK          0x1Fu
#define ADC_SC1_ADCH_SHIFT         0
#define ADC_SC1_ADCH_WIDTH         5
#define ADC_SC1_ADCH(x)            (((uint32_t)((uint32_t)(x)<<ADC_SC1_ADCH_SHIFT))&ADC_SC1_ADCH_MASK)

#define ADC_SC1_DIFF_MASK          0x20u
#define ADC_SC1_DIFF_SHIFT         5
#define ADC_SC1_DIFF_WIDTH         1
#define ADC_SC1_DIFF(x)            (((uint32_t)((uint32_t)(x)<<ADC_SC1_DIFF_SHIFT))&ADC_SC1_DIFF_MASK)

#define ADC_SC1_AIEN_MASK          0x40u
#define ADC_SC1_AIEN_SHIFT         6
#define ADC_SC1_AIEN_WIDTH         1
#define ADC_SC1_AIEN(x)            (((uint32_t)((uint32_t)(x)<<ADC_SC1_AIEN_SHIFT))&ADC_SC1_AIEN_MASK)

#define ADC_SC1_COCO_MASK          0x80u
#define ADC_SC1_COCO_SHIFT         7
#define ADC_SC1_COCO_WIDTH         1
#define ADC_SC1_COCO(x)            (((uint32_t)((uint32_t)(x)<<ADC_SC1_COCO_SHIFT))&ADC_SC1_COCO_MASK)

/* CFG1 Bit Fields */

#define ADC_CFG1_ADICLK_MASK        0x3u
#define ADC_CFG1_ADICLK_SHIFT       0
#define ADC_CFG1_ADICLK_WIDTH       2
#define ADC_CFG1_ADICLK(x)          (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADICLK_SHIFT))&ADC_CFG1_ADICLK_MASK)

#define ADC_CFG1_MODE_MASK          0xCu
```

```

#define ADC_CFG1_MODE_SHIFT          2
#define ADC_CFG1_MODE_WIDTH          2
#define ADC_CFG1_MODE(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG1_MODE_SHIFT))&ADC_CFG1_MODE_MASK)

#define ADC_CFG1_ADLSMP_MASK         0x10u
#define ADC_CFG1_ADLSMP_SHIFT        4
#define ADC_CFG1_ADLSMP_WIDTH        1
#define ADC_CFG1_ADLSMP(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADLSMP_SHIFT))&ADC_CFG1_ADLSMP_MASK)

#define ADC_CFG1_ADIV_MASK           0x60u
#define ADC_CFG1_ADIV_SHIFT          5
#define ADC_CFG1_ADIV_WIDTH          2
#define ADC_CFG1_ADIV(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADIV_SHIFT))&ADC_CFG1_ADIV_MASK)

#define ADC_CFG1_ADLPC_MASK          0x80u
#define ADC_CFG1_ADLPC_SHIFT         7
#define ADC_CFG1_ADLPC_WIDTH         1
#define ADC_CFG1_ADLPC(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADLPC_SHIFT))&ADC_CFG1_ADLPC_MASK)

/* CFG2 Bit Fields */

#define ADC_CFG2_DLSTS_MASK          0x3u
#define ADC_CFG2_DLSTS_SHIFT          0
#define ADC_CFG2_DLSTS_WIDTH          2
#define ADC_CFG2_DLSTS(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG2_DLSTS_SHIFT))&ADC_CFG2_DLSTS_MASK)

#define ADC_CFG2_ADHSC_MASK           0x4u
#define ADC_CFG2_ADHSC_SHIFT          2
#define ADC_CFG2_ADHSC_WIDTH          1
#define ADC_CFG2_ADHSC(x)
(((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADHSC_SHIFT))&ADC_CFG2_ADHSC_MASK)

#define ADC_CFG2_ADACKEN_MASK         0x8u

```

```

#define ADC_CFG2_ADACKEN_SHIFT          3
#define ADC_CFG2_ADACKEN_WIDTH          1
#define ADC_CFG2_ADACKEN(x)             (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADACKEN_SHIFT))&ADC_CFG2_ADACKEN_MASK)

#define ADC_CFG2_MUXSEL_MASK            0x10u
#define ADC_CFG2_MUXSEL_SHIFT           4
#define ADC_CFG2_MUXSEL_WIDTH           1
#define ADC_CFG2_MUXSEL(x)              (((uint32_t)((uint32_t)(x)<<ADC_CFG2_MUXSEL_SHIFT))&ADC_CFG2_MUXSEL_MASK)

/* R Bit Fields */

#define ADC_R_D_MASK                  0xFFFFu
#define ADC_R_D_SHIFT                 0
#define ADC_R_D_WIDTH                 16
#define ADC_R_D(x)                    (((uint32_t)((uint32_t)(x)<<ADC_R_D_SHIFT))&ADC_R_D_MASK)

/* CV1 Bit Fields */

#define ADC.CV1.CV_MASK                0xFFFFu
#define ADC.CV1.CV_SHIFT               0
#define ADC.CV1.CV_WIDTH               16
#define ADC.CV1.CV(x)                 (((uint32_t)((uint32_t)(x)<<ADC.CV1.CV_SHIFT))&ADC.CV1.CV_MASK)

/* CV2 Bit Fields */

#define ADC.CV2.CV_MASK                0xFFFFu
#define ADC.CV2.CV_SHIFT               0
#define ADC.CV2.CV_WIDTH               16
#define ADC.CV2.CV(x)                 (((uint32_t)((uint32_t)(x)<<ADC.CV2.CV_SHIFT))&ADC.CV2.CV_MASK)

/* SC2 Bit Fields */

#define ADC.SC2.REFSEL_MASK             0x3u
#define ADC.SC2.REFSEL_SHIFT            0
#define ADC.SC2.REFSEL_WIDTH            2

```

```

#define ADC_SC2_REFSEL(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_REFSEL_SHIFT))&ADC_SC2_REFSEL_MASK)

#define ADC_SC2_DMAEN_MASK          0x4u
#define ADC_SC2_DMAEN_SHIFT        2
#define ADC_SC2_DMAEN_WIDTH        1

#define ADC_SC2_DMAEN(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_DMAEN_SHIFT))&ADC_SC2_DMAEN_MASK)

#define ADC_SC2_ACREN_MASK          0x8u
#define ADC_SC2_ACREN_SHIFT        3
#define ADC_SC2_ACREN_WIDTH        1

#define ADC_SC2_ACREN(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACREN_SHIFT))&ADC_SC2_ACREN_MASK)

#define ADC_SC2_ACFGTE_MASK         0x10u
#define ADC_SC2_ACFGTE_SHIFT       4
#define ADC_SC2_ACFGTE_WIDTH       1

#define ADC_SC2_ACFGTE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFGTE_SHIFT))&ADC_SC2_ACFGTE_MASK)

#define ADC_SC2_ACFE_MASK           0x20u
#define ADC_SC2_ACFE_SHIFT         5
#define ADC_SC2_ACFE_WIDTH         1

#define ADC_SC2_ACFE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFE_SHIFT))&ADC_SC2_ACFE_MASK)

#define ADC_SC2_ADTRG_MASK          0x40u
#define ADC_SC2_ADTRG_SHIFT        6
#define ADC_SC2_ADTRG_WIDTH        1

#define ADC_SC2_ADTRG(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADTRG_SHIFT))&ADC_SC2_ADTRG_MASK)

#define ADC_SC2_ADACT_MASK          0x80u
#define ADC_SC2_ADACT_SHIFT        7
#define ADC_SC2_ADACT_WIDTH        1

```

```

#define ADC_SC2_ADACT(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADACT_SHIFT))&ADC_SC2_ADACT_MASK)

/* SC3 Bit Fields */

#define ADC_SC3_AVGS_MASK          0x3u
#define ADC_SC3_AVGS_SHIFT         0
#define ADC_SC3_AVGS_WIDTH         2

#define ADC_SC3_AVGS(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGS_SHIFT))&ADC_SC3_AVGS_MASK)

#define ADC_SC3_AVGE_MASK          0x4u
#define ADC_SC3_AVGE_SHIFT         2
#define ADC_SC3_AVGE_WIDTH         1

#define ADC_SC3_AVGE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGE_SHIFT))&ADC_SC3_AVGE_MASK)

#define ADC_SC3_ADCO_MASK          0x8u
#define ADC_SC3_ADCO_SHIFT         3
#define ADC_SC3_ADCO_WIDTH         1

#define ADC_SC3_ADCO(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_ADCO_SHIFT))&ADC_SC3_ADCO_MASK)

#define ADC_SC3_CALF_MASK          0x40u
#define ADC_SC3_CALF_SHIFT         6
#define ADC_SC3_CALF_WIDTH         1

#define ADC_SC3_CALF(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CALF_SHIFT))&ADC_SC3_CALF_MASK)

#define ADC_SC3_CAL_MASK           0x80u
#define ADC_SC3_CAL_SHIFT          7
#define ADC_SC3_CAL_WIDTH          1

#define ADC_SC3_CAL(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CAL_SHIFT))&ADC_SC3_CAL_MASK)

/* OFS Bit Fields */

#define ADC_OFS_OFS_MASK           0xFFFFu
#define ADC_OFS_OFS_SHIFT          0

```

```

#define ADC_OFS_OFS_WIDTH          16
#define ADC_OFS_OFS(x)             (((uint32_t)((uint32_t)(x))<<ADC_OFS_OFS_SHIFT))&ADC_OFS_OFS_MASK)

/* PG Bit Fields */

#define ADC_PG_PG_MASK            0xFFFFu
#define ADC_PG_PG_SHIFT           0
#define ADC_PG_PG_WIDTH           16
#define ADC_PG_PG(x)              (((uint32_t)((uint32_t)(x))<<ADC_PG_PG_SHIFT))&ADC_PG_PG_MASK)

/* MG Bit Fields */

#define ADC_MG_MG_MASK            0xFFFFu
#define ADC_MG_MG_SHIFT           0
#define ADC_MG_MG_WIDTH           16
#define ADC_MG_MG(x)              (((uint32_t)((uint32_t)(x))<<ADC_MG_MG_SHIFT))&ADC_MG_MG_MASK)

/* CLPD Bit Fields */

#define ADC_CLPD_CLPD_MASK         0x3Fu
#define ADC_CLPD_CLPD_SHIFT        0
#define ADC_CLPD_CLPD_WIDTH        6
#define ADC_CLPD_CLPD(x)           (((uint32_t)((uint32_t)(x))<<ADC_CLPD_CLPD_SHIFT))&ADC_CLPD_CLPD_MASK)

/* CLPS Bit Fields */

#define ADC_CLPS_CLPS_MASK         0x3Fu
#define ADC_CLPS_CLPS_SHIFT        0
#define ADC_CLPS_CLPS_WIDTH        6
#define ADC_CLPS_CLPS(x)           (((uint32_t)((uint32_t)(x))<<ADC_CLPS_CLPS_SHIFT))&ADC_CLPS_CLPS_MASK)

/* CLP4 Bit Fields */

#define ADC_CLP4_CLP4_MASK         0x3FFu
#define ADC_CLP4_CLP4_SHIFT        0

```

```

#define ADC_CLP4_CLP4_WIDTH          10
#define ADC_CLP4_CLP4(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP4_CLP4_SHIFT))&ADC_CLP4_CLP4_MASK)

/* CLP3 Bit Fields */

#define ADC_CLP3_CLP3_MASK           0x1FFu
#define ADC_CLP3_CLP3_SHIFT          0
#define ADC_CLP3_CLP3_WIDTH          9
#define ADC_CLP3_CLP3(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP3_CLP3_SHIFT))&ADC_CLP3_CLP3_MASK)

/* CLP2 Bit Fields */

#define ADC_CLP2_CLP2_MASK           0xFFu
#define ADC_CLP2_CLP2_SHIFT          0
#define ADC_CLP2_CLP2_WIDTH          8
#define ADC_CLP2_CLP2(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP2_CLP2_SHIFT))&ADC_CLP2_CLP2_MASK)

/* CLP1 Bit Fields */

#define ADC_CLP1_CLP1_MASK           0x7Fu
#define ADC_CLP1_CLP1_SHIFT          0
#define ADC_CLP1_CLP1_WIDTH          7
#define ADC_CLP1_CLP1(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP1_CLP1_SHIFT))&ADC_CLP1_CLP1_MASK)

/* CLP0 Bit Fields */

#define ADC_CLP0_CLP0_MASK           0x3Fu
#define ADC_CLP0_CLP0_SHIFT          0
#define ADC_CLP0_CLP0_WIDTH          6
#define ADC_CLP0_CLP0(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP0_CLP0_SHIFT))&ADC_CLP0_CLP0_MASK)

/* CLMD Bit Fields */

#define ADC_CLMD_CLMD_MASK           0x3Fu
#define ADC_CLMD_CLMD_SHIFT          0

```

```

#define ADC_CLMD_CLMD_WIDTH          6
#define ADC_CLMD_CLMD(x) (((uint32_t)((uint32_t)(x)<<ADC_CLMD_CLMD_SHIFT))&ADC_CLMD_CLMD_MASK)

/* CLMS Bit Fields */

#define ADC_CLMS_CLMS_MASK           0x3Fu
#define ADC_CLMS_CLMS_SHIFT          0
#define ADC_CLMS_CLMS_WIDTH          6
#define ADC_CLMS_CLMS(x) (((uint32_t)((uint32_t)(x)<<ADC_CLMS_CLMS_SHIFT))&ADC_CLMS_CLMS_MASK)

/* CLM4 Bit Fields */

#define ADC_CLM4_CLM4_MASK           0x3FFu
#define ADC_CLM4_CLM4_SHIFT          0
#define ADC_CLM4_CLM4_WIDTH          10
#define ADC_CLM4_CLM4(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM4_CLM4_SHIFT))&ADC_CLM4_CLM4_MASK)

/* CLM3 Bit Fields */

#define ADC_CLM3_CLM3_MASK           0x1FFu
#define ADC_CLM3_CLM3_SHIFT          0
#define ADC_CLM3_CLM3_WIDTH          9
#define ADC_CLM3_CLM3(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM3_CLM3_SHIFT))&ADC_CLM3_CLM3_MASK)

/* CLM2 Bit Fields */

#define ADC_CLM2_CLM2_MASK           0xFFu
#define ADC_CLM2_CLM2_SHIFT          0
#define ADC_CLM2_CLM2_WIDTH          8
#define ADC_CLM2_CLM2(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM2_CLM2_SHIFT))&ADC_CLM2_CLM2_MASK)

/* CLM1 Bit Fields */

#define ADC_CLM1_CLM1_MASK           0x7Fu
#define ADC_CLM1_CLM1_SHIFT          0

```

```

#define ADC_CLM1_CLM1_WIDTH          7
#define ADC_CLM1_CLM1(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM1_CLM1_SHIFT))&ADC_CLM1_CLM1_MASK)
/* CLM0 Bit Fields */
#define ADC_CLM0_CLM0_MASK           0x3Fu
#define ADC_CLM0_CLM0_SHIFT          0
#define ADC_CLM0_CLM0_WIDTH          6
#define ADC_CLM0_CLM0(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM0_CLM0_SHIFT))&ADC_CLM0_CLM0_MASK)

/*
 * @}
 */ /* end of group ADC_Register_Masks */

```

```

/* ADC - Peripheral instance base addresses */
/** Peripheral ADC0 base address */
#define ADC0_BASE                   (0x4003B000u)
/** Peripheral ADC0 base pointer */
#define ADC0 ((ADC_Type *)ADC0_BASE)
#define ADC0_BASE_PTR               (ADC0)
/** Array initializer of ADC peripheral base addresses */
#define ADC_BASE_ADDRS              { ADC0_BASE }
/** Array initializer of ADC peripheral base pointers */
#define ADC_BASE_PTRS                { ADC0 }

```

```

/*
-- ADC - Register accessor macros
----- */

```

```

/*!

* @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros

* @{

*/



/* ADC - Register instance definitions */

/* ADC0 */

#define ADC0_SC1A           ADC_SC1_REG(ADC0,0)
#define ADC0_SC1B           ADC_SC1_REG(ADC0,1)
#define ADC0_CFG1            ADC_CFG1_REG(ADC0)
#define ADC0_CFG2            ADC_CFG2_REG(ADC0)
#define ADC0_RA              ADC_R_REG(ADC0,0)
#define ADC0_RB              ADC_R_REG(ADC0,1)
#define ADC0.CV1             ADC_CV1_REG(ADC0)
#define ADC0.CV2             ADC_CV2_REG(ADC0)
#define ADC0_SC2            ADC_SC2_REG(ADC0)
#define ADC0_SC3            ADC_SC3_REG(ADC0)
#define ADC0_OFS             ADC_OFS_REG(ADC0)
#define ADC0_PG              ADC_PG_REG(ADC0)
#define ADC0_MG              ADC_MG_REG(ADC0)
#define ADC0_CLPD            ADC_CLPD_REG(ADC0)
#define ADC0_CLPS            ADC_CLPS_REG(ADC0)
#define ADC0_CLP4            ADC_CLP4_REG(ADC0)
#define ADC0_CLP3            ADC_CLP3_REG(ADC0)
#define ADC0_CLP2            ADC_CLP2_REG(ADC0)
#define ADC0_CLP1            ADC_CLP1_REG(ADC0)
#define ADC0_CLP0            ADC_CLP0_REG(ADC0)

```

```
#define ADC0_CLMD          ADC_CLMD_REG(ADC0)
#define ADC0_CLMS          ADC_CLMS_REG(ADC0)
#define ADC0_CLM4          ADC_CLM4_REG(ADC0)
#define ADC0_CLM3          ADC_CLM3_REG(ADC0)
#define ADC0_CLM2          ADC_CLM2_REG(ADC0)
#define ADC0_CLM1          ADC_CLM1_REG(ADC0)
#define ADC0_CLM0          ADC_CLM0_REG(ADC0)
```

```
/* ADC - Register array accessors */

#define ADC0_SC1(index)      ADC_SC1_REG(ADC0,index)
#define ADC0_R(index)        ADC_R_REG(ADC0,index)
```

```
/*
 * @}
 */
/* end of group ADC_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group ADC_Peripheral_Access_Layer */
```

```
/*
-----  
-- CMP Peripheral Access Layer  
----- */
```

```
/*
 * @addtogroup CMP_Peripheral_Access_Layer CMP Peripheral Access Layer
 * @{
 */
```

```
 */
```

```
/** CMP - Register Layout Typedef */

typedef struct {

    __IO uint8_t CRO;           /**< CMP Control Register 0, offset: 0x0 */
    __IO uint8_t CR1;           /**< CMP Control Register 1, offset: 0x1 */
    __IO uint8_t FPR;           /**< CMP Filter Period Register, offset: 0x2 */
    __IO uint8_t SCR;           /**< CMP Status and Control Register, offset: 0x3 */
    __IO uint8_t DACCR;         /**< DAC Control Register, offset: 0x4 */
    __IO uint8_t MUXCR;         /**< MUX Control Register, offset: 0x5 */

} CMP_Type, *CMP_MemMapPtr;
```

```
/*
-----  
-- CMP - Register accessor macros  
----- */
```

```
/*!
```

```
* @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros
* @{
* /
```

```
/* CMP - Register accessors */
```

```
#define CMP_CRO_REG(base)      ((base)->CRO)
#define CMP_CR1_REG(base)        ((base)->CR1)
#define CMP_FPR_REG(base)        ((base)->FPR)
#define CMP_SCR_REG(base)        ((base)->SCR)
#define CMP_DACCR_REG(base)      ((base)->DACCR)
#define CMP_MUXCR_REG(base)      ((base)->MUXCR)
```

```
/*!  
 * @}  
 */ /* end of group CMP_Register_Accessor_Macros */
```

```
/* -----  
-- CMP Register Masks  
----- */
```

```
/*!  
 * @addtogroup CMP_Register_Masks CMP Register Masks  
 * @{  
 */
```

```
/* CRO Bit Fields */  
  
#define CMP_CRO_HYSTCTR_MASK          0x3u  
#define CMP_CRO_HYSTCTR_SHIFT         0  
#define CMP_CRO_HYSTCTR_WIDTH         2  
  
#define CMP_CRO_HYSTCTR(x)  
(((uint8_t)((uint8_t)(x)<<CMP_CRO_HYSTCTR_SHIFT))&CMP_CRO_HYSTCTR_MASK)  
  
#define CMP_CRO_FILTER_CNT_MASK        0x70u  
#define CMP_CRO_FILTER_CNT_SHIFT       4  
#define CMP_CRO_FILTER_CNT_WIDTH       3  
  
#define CMP_CRO_FILTER_CNT(x)  
(((uint8_t)((uint8_t)(x)<<CMP_CRO_FILTER_CNT_SHIFT))&CMP_CRO_FILTER_CNT_MASK)  
  
/* CR1 Bit Fields */  
  
#define CMP_CR1_EN_MASK               0x1u  
#define CMP_CR1_EN_SHIFT              0
```

```

#define CMP_CR1_EN_WIDTH          1
#define CMP_CR1_EN(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_EN_SHIFT))&CMP_CR1_EN_MASK)
#define CMP_CR1_OPE_MASK          0x2u
#define CMP_CR1_OPE_SHIFT         1
#define CMP_CR1_OPE_WIDTH         1
#define CMP_CR1_OPE(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_OPE_SHIFT))&CMP_CR1_OPE_MASK)
#define CMP_CR1_COS_MASK          0x4u
#define CMP_CR1_COS_SHIFT         2
#define CMP_CR1_COS_WIDTH         1
#define CMP_CR1_COS(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_COS_SHIFT))&CMP_CR1_COS_MASK)
#define CMP_CR1_INV_MASK           0x8u
#define CMP_CR1_INV_SHIFT          3
#define CMP_CR1_INV_WIDTH          1
#define CMP_CR1_INV(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_INV_SHIFT))&CMP_CR1_INV_MASK)
#define CMP_CR1_PMODE_MASK          0x10u
#define CMP_CR1_PMODE_SHIFT         4
#define CMP_CR1_PMODE_WIDTH         1
#define CMP_CR1_PMODE(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_PMODE_SHIFT))&CMP_CR1_PMODE_MASK)
#define CMP_CR1_TRIGM_MASK          0x20u
#define CMP_CR1_TRIGM_SHIFT          5
#define CMP_CR1_TRIGM_WIDTH          1
#define CMP_CR1_TRIGM(x) (((uint8_t)((uint8_t)(x)<<CMP_CR1_TRIGM_SHIFT))&CMP_CR1_TRIGM_MASK)
#define CMP_CR1_WE_MASK             0x40u
#define CMP_CR1_WE_SHIFT              6
#define CMP_CR1_WE_WIDTH              1

```

```

#define CMP_CR1_WE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_WE_SHIFT))&CMP_CR1_WE_MASK)

#define CMP_CR1_SE_MASK          0x80u
#define CMP_CR1_SE_SHIFT         7
#define CMP_CR1_SE_WIDTH         1

#define CMP_CR1_SE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_SE_SHIFT))&CMP_CR1_SE_MASK)

/* FPR Bit Fields */

#define CMP_FPR_FILT_PER_MASK    0xFFu
#define CMP_FPR_FILT_PER_SHIFT   0
#define CMP_FPR_FILT_PER_WIDTH   8

#define CMP_FPR_FILT_PER(x)
(((uint8_t) (((uint8_t)(x))<<CMP_FPR_FILT_PER_SHIFT))&CMP_FPR_FILT_PER_MASK)

/* SCR Bit Fields */

#define CMP_SCR_COUT_MASK        0x1u
#define CMP_SCR_COUT_SHIFT       0
#define CMP_SCR_COUT_WIDTH       1

#define CMP_SCR_COUT(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_COUT_SHIFT))&CMP_SCR_COUT_MASK)

#define CMP_SCR_CFF_MASK         0x2u
#define CMP_SCR_CFF_SHIFT        1
#define CMP_SCR_CFF_WIDTH        1

#define CMP_SCR_CFF(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_CFF_SHIFT))&CMP_SCR_CFF_MASK)

#define CMP_SCR_CFR_MASK         0x4u
#define CMP_SCR_CFR_SHIFT        2
#define CMP_SCR_CFR_WIDTH        1

#define CMP_SCR_CFR(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_CFR_SHIFT))&CMP_SCR_CFR_MASK)

#define CMP_SCR_IEF_MASK         0x8u
#define CMP_SCR_IEF_SHIFT        3

```

```

#define CMP_SCR_IEF_WIDTH          1
#define CMP_SCR_IEF(x) (((uint8_t)((uint8_t)(x)<<CMP_SCR_IEF_SHIFT))&CMP_SCR_IEF_MASK)
#define CMP_SCR_IER_MASK           0x10u
#define CMP_SCR_IER_SHIFT          4
#define CMP_SCR_IER_WIDTH          1
#define CMP_SCR_IER(x) (((uint8_t)((uint8_t)(x)<<CMP_SCR_IER_SHIFT))&CMP_SCR_IER_MASK)
#define CMP_SCR_DMAEN_MASK         0x40u
#define CMP_SCR_DMAEN_SHIFT        6
#define CMP_SCR_DMAEN_WIDTH        1
#define CMP_SCR_DMAEN(x) (((uint8_t)((uint8_t)(x)<<CMP_SCR_DMAEN_SHIFT))&CMP_SCR_DMAEN_MASK)

/* DACC Bit Fields */

#define CMP_DACCR_VOSEL_MASK       0x3Fu
#define CMP_DACCR_VOSEL_SHIFT      0
#define CMP_DACCR_VOSEL_WIDTH      6
#define CMP_DACCR_VOSEL(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_VOSEL_SHIFT))&CMP_DACCR_VOSEL_MASK)
#define CMP_DACCR_VRSEL_MASK        0x40u
#define CMP_DACCR_VRSEL_SHIFT       6
#define CMP_DACCR_VRSEL_WIDTH       1
#define CMP_DACCR_VRSEL(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_VRSEL_SHIFT))&CMP_DACCR_VRSEL_MASK)
#define CMP_DACCR_DACEN_MASK        0x80u
#define CMP_DACCR_DACEN_SHIFT       7
#define CMP_DACCR_DACEN_WIDTH       1
#define CMP_DACCR_DACEN(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_DACEN_SHIFT))&CMP_DACCR_DACEN_MASK)

/* MUXCR Bit Fields */

#define CMP_MUXCR_MSEL_MASK         0x7u

```

```

#define CMP_MUXCR_MSEL_SHIFT          0
#define CMP_MUXCR_MSEL_WIDTH          3
#define CMP_MUXCR_MSEL(x)             (((uint8_t)((uint8_t)(x)<<CMP_MUXCR_MSEL_SHIFT))&CMP_MUXCR_MSEL_MASK)
#define CMP_MUXCR_PSEL_MASK           0x38u
#define CMP_MUXCR_PSEL_SHIFT          3
#define CMP_MUXCR_PSEL_WIDTH          3
#define CMP_MUXCR_PSEL(x)             (((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSEL_SHIFT))&CMP_MUXCR_PSEL_MASK)
#define CMP_MUXCR_PSTM_MASK           0x80u
#define CMP_MUXCR_PSTM_SHIFT          7
#define CMP_MUXCR_PSTM_WIDTH          1
#define CMP_MUXCR_PSTM(x)             (((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSTM_SHIFT))&CMP_MUXCR_PSTM_MASK)

```

```

/*
 * @}
 */
/* end of group CMP_Register_Masks */

```

```

/* CMP - Peripheral instance base addresses */
/** Peripheral CMPO base address */
#define CMPO_BASE                   (0x40073000u)
/** Peripheral CMPO base pointer */
#define CMPO                         ((CMP_Type *)CMPO_BASE)
#define CMPO_BASE_PTR                (CMPO)
/** Array initializer of CMP peripheral base addresses */
#define CMP_BASE_ADDRS               { CMPO_BASE }
/** Array initializer of CMP peripheral base pointers */
#define CMP_BASE_PTRS                { CMPO }

```

```

/*
-----  

-- CMP - Register accessor macros  

----- */
  

/*!  

 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros  

 * @{
 * /  

  

/* CMP - Register instance definitions */  

/* CMPO */  

#define CMPO_CRO           CMP_CRO_REG(CMPO)  

#define CMPO_CR1           CMP_CR1_REG(CMPO)  

#define CMPO_FPR           CMP_FPR_REG(CMPO)  

#define CMPO_SCR           CMP_SCR_REG(CMPO)  

#define CMPO_DACCR         CMP_DACCR_REG(CMPO)  

#define CMPO_MUXCR         CMP_MUXCR_REG(CMPO)  

  

/*!  

 * @}  

 * /* end of group CMP_Register_Accessor_Macros */  

  

/*!  

 * @}  

/* /* end of group CMP_Peripheral_Access_Layer */

```

```

/*
-----*
-- DAC Peripheral Access Layer
----- */
/*!
* @addtogroup DAC_Peripheral_Access_Layer DAC Peripheral Access Layer
* @{
*/
/* ** DAC - Register Layout Typedef */
typedef struct {
    struct {                                     /* offset: 0x0, array step: 0x2 */
        __IO uint8_t DATL;                      /* **< DAC Data Low Register, array offset: 0x0, array step: 0x2 */
        __IO uint8_t DATH;                      /* **< DAC Data High Register, array offset: 0x1, array step: 0x2 */
    } DAT[2];
    uint8_t RESERVED_0[28];
    __IO uint8_t SR;                          /* **< DAC Status Register, offset: 0x20 */
    __IO uint8_t C0;                          /* **< DAC Control Register, offset: 0x21 */
    __IO uint8_t C1;                          /* **< DAC Control Register 1, offset: 0x22 */
    __IO uint8_t C2;                          /* **< DAC Control Register 2, offset: 0x23 */
} DAC_Type, *DAC_MemMapPtr;

/*
-----*
-- DAC - Register accessor macros
----- */
/*!
* @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros
*/

```

```
* @{

/*
 * DAC - Register accessors */

#define DAC_DATL_REG(base,index)      ((base)->DAT[index].DATL)
#define DAC_DATL_COUNT                2
#define DAC_DATH_REG(base,index)      ((base)->DAT[index].DATH)
#define DAC_DATH_COUNT                2
#define DAC_SR_REG(base)              ((base)->SR)
#define DAC_C0_REG(base)              ((base)->C0)
#define DAC_C1_REG(base)              ((base)->C1)
#define DAC_C2_REG(base)              ((base)->C2)

/*!
 * @}

 */ /* end of group DAC_Register_Accessor_Macros */
```

```
/*
-----  
-- DAC Register Masks  
----- */

/*!
```

```
* @addtogroup DAC_Register_Masks DAC Register Masks
* @{
*/
```

```
/* DATL Bit Fields */
```

```

#define DAC_DATL_DATA0_MASK          0xFFu
#define DAC_DATL_DATA0_SHIFT         0
#define DAC_DATL_DATA0_WIDTH         8
#define DAC_DATL_DATA0(x) (((uint8_t)((uint8_t)(x))<<DAC_DATL_DATA0_SHIFT))&DAC_DATL_DATA0_MASK)

/* DATH Bit Fields */

#define DAC_DATH_DATA1_MASK          0xFu
#define DAC_DATH_DATA1_SHIFT         0
#define DAC_DATH_DATA1_WIDTH         4
#define DAC_DATH_DATA1(x) (((uint8_t)((uint8_t)(x))<<DAC_DATH_DATA1_SHIFT))&DAC_DATH_DATA1_MASK)

/* SR Bit Fields */

#define DAC_SR_DACBFRPBF_MASK        0x1u
#define DAC_SR_DACBFRPBF_SHIFT       0
#define DAC_SR_DACBFRPBF_WIDTH       1
#define DAC_SR_DACBFRPBF(x) (((uint8_t)((uint8_t)(x))<<DAC_SR_DACBFRPBF_SHIFT))&DAC_SR_DACBFRPBF_MASK)

#define DAC_SR_DACBFRPTF_MASK        0x2u
#define DAC_SR_DACBFRPTF_SHIFT       1
#define DAC_SR_DACBFRPTF_WIDTH       1
#define DAC_SR_DACBFRPTF(x) (((uint8_t)((uint8_t)(x))<<DAC_SR_DACBFRPTF_SHIFT))&DAC_SR_DACBFRPTF_MASK)

/* C0 Bit Fields */

#define DAC_C0_DACBBIEN_MASK         0x1u
#define DAC_C0_DACBBIEN_SHIFT        0
#define DAC_C0_DACBBIEN_WIDTH        1
#define DAC_C0_DACBBIEN(x) (((uint8_t)((uint8_t)(x))<<DAC_C0_DACBBIEN_SHIFT))&DAC_C0_DACBBIEN_MASK)

#define DAC_C0_DACBTIEN_MASK         0x2u
#define DAC_C0_DACBTIEN_SHIFT        1

```

```

#define DAC_C0_DACBTIEN_WIDTH          1
#define DAC_C0_DACBTIEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACBTIEN_SHIFT))&DAC_C0_DACBTIEN_MASK)
#define DAC_C0_LPEN_MASK               0x8u
#define DAC_C0_LPEN_SHIFT              3
#define DAC_C0_LPEN_WIDTH              1
#define DAC_C0_LPEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_LPEN_SHIFT))&DAC_C0_LPEN_MASK)
#define DAC_C0_DACSWTRG_MASK           0x10u
#define DAC_C0_DACSWTRG_SHIFT          4
#define DAC_C0_DACSWTRG_WIDTH          1
#define DAC_C0_DACSWTRG(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACSWTRG_SHIFT))&DAC_C0_DACSWTRG_MASK)
#define DAC_C0_DACTRGSEL_MASK          0x20u
#define DAC_C0_DACTRGSEL_SHIFT         5
#define DAC_C0_DACTRGSEL_WIDTH         1
#define DAC_C0_DACTRGSEL(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACTRGSEL_SHIFT))&DAC_C0_DACTRGSEL_MASK)
#define DAC_C0_DACRFS_MASK             0x40u
#define DAC_C0_DACRFS_SHIFT            6
#define DAC_C0_DACRFS_WIDTH            1
#define DAC_C0_DACRFS(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACRFS_SHIFT))&DAC_C0_DACRFS_MASK)
#define DAC_C0_DACEN_MASK              0x80u
#define DAC_C0_DACEN_SHIFT             7
#define DAC_C0_DACEN_WIDTH             1
#define DAC_C0_DACEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACEN_SHIFT))&DAC_C0_DACEN_MASK)
/* C1 Bit Fields */
#define DAC_C1_DACBFEN_MASK            0x1u
#define DAC_C1_DACBFEN_SHIFT           0

```

```

#define DAC_C1_DACBFEN_WIDTH          1
#define DAC_C1_DACBFEN(x)             (((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFEN_SHIFT))&DAC_C1_DACBFEN_MASK)
#define DAC_C1_DACBFMD_MASK           0x4u
#define DAC_C1_DACBFMD_SHIFT          2
#define DAC_C1_DACBFMD_WIDTH          1
#define DAC_C1_DACBFMD(x)             (((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFMD_SHIFT))&DAC_C1_DACBFMD_MASK)
#define DAC_C1_DMAEN_MASK             0x80u
#define DAC_C1_DMAEN_SHIFT            7
#define DAC_C1_DMAEN_WIDTH            1
#define DAC_C1_DMAEN(x)               (((uint8_t)((uint8_t)(x)<<DAC_C1_DMAEN_SHIFT))&DAC_C1_DMAEN_MASK)

/* C2 Bit Fields */

#define DAC_C2_DACBFUP_MASK           0x1u
#define DAC_C2_DACBFUP_SHIFT          0
#define DAC_C2_DACBFUP_WIDTH          1
#define DAC_C2_DACBFUP(x)              (((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFUP_SHIFT))&DAC_C2_DACBFUP_MASK)
#define DAC_C2_DACBFRP_MASK           0x10u
#define DAC_C2_DACBFRP_SHIFT          4
#define DAC_C2_DACBFRP_WIDTH          1
#define DAC_C2_DACBFRP(x)              (((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFRP_SHIFT))&DAC_C2_DACBFRP_MASK)

/*
 * @}
 */
/* /* end of group DAC_Register_Masks */

```

```

/* DAC - Peripheral instance base addresses */

/** Peripheral DAC0 base address */

#define DAC0_BASE          (0x4003F000u)

/** Peripheral DAC0 base pointer */

#define DAC0           ((DAC_Type *)DAC0_BASE)

#define DAC0_BASE_PTR      (DAC0)

/** Array initializer of DAC peripheral base addresses */

#define DAC_BASE_ADDRS     { DAC0_BASE }

/** Array initializer of DAC peripheral base pointers */

#define DAC_BASE_PTRS      { DAC0 }

```

```

/*
-----  

-- DAC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros  

* @{
* /  


```

```

/* DAC - Register instance definitions */

/* DAC0 */

#define DAC0_DAT0L        DAC_DATL_REG(DAC0,0)  

#define DAC0_DAT0H        DAC_DATH_REG(DAC0,0)  

#define DAC0_DAT1L        DAC_DATL_REG(DAC0,1)  

#define DAC0_DAT1H        DAC_DATH_REG(DAC0,1)  

#define DAC0_SR           DAC_SR_REG(DAC0)  

#define DAC0_CO           DAC_CO_REG(DAC0)

```

```
#define DAC0_C1           DAC_C1_REG(DAC0)
#define DAC0_C2           DAC_C2_REG(DAC0)

/* DAC - Register array accessors */

#define DAC0_DATL(index)      DAC_DATL_REG(DAC0,index)
#define DAC0_DATH(index)      DAC_DATH_REG(DAC0,index)
```

```
/*
 * @}
 */
/* /* end of group DAC_Register_Accessor_Macros */
```

```
/*
 * @}
*/
/* /* end of group DAC_Peripheral_Access_Layer */
```

```
/*
-----  
-- DMA Peripheral Access Layer  
----- */  
  
/*
 * @addtogroup DMA_Peripheral_Access_Layer DMA Peripheral Access Layer
 * @{
 */

```

```
/** DMA - Register Layout Typedef */  
  
typedef struct {  
    uint8_t RESERVED_0[256];
```

```

struct {                                     /* offset: 0x100, array step: 0x10 */
    __IO uint32_t SAR;                      /* **< Source Address Register, array offset: 0x100, array step:
0x10 */
    __IO uint32_t DAR;                      /* **< Destination Address Register, array offset: 0x104, array step:
0x10 */
    union {                                 /* offset: 0x108, array step: 0x10 */
        __IO uint32_t DSR_BCR;               /* **< DMA Status Register / Byte Count Register, array offset:
0x108, array step: 0x10 */
        struct {                           /* offset: 0x108, array step: 0x10 */
            uint8_t RESERVED_0[3];
            __IO uint8_t DSR;                 /* **< DMA_DSR0 register...DMA_DSR3 register., array offset:
0x10B, array step: 0x10 */
        } DMA_DSR_ACCESS8BIT;
    };
    __IO uint32_t DCR;                      /* **< DMA Control Register, array offset: 0x10C, array step: 0x10
*/
} DMA[4];
} DMA_Type, *DMA_MemMapPtr;

/* -----
-- DMA - Register accessor macros
----- */

/*!
* @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros
* @{
*/
/* DMA - Register accessors */

```

```

#define DMA_SAR_REG(base,index)      ((base)->DMA[index].SAR)
#define DMA_SAR_COUNT              4
#define DMA_DAR_REG(base,index)     ((base)->DMA[index].DAR)
#define DMA_DAR_COUNT              4
#define DMA_DSR_BCR_REG(base,index) ((base)->DMA[index].DSR_BCR)
#define DMA_DSR_BCR_COUNT          4
#define DMA_DSR_REG(base,index)     ((base)->DMA[index].DMA_DSR_ACCESS8BIT.DSR)
#define DMA_DSR_COUNT              4
#define DMA_DCR_REG(base,index)     ((base)->DMA[index].DCR)
#define DMA_DCR_COUNT              4

/*!
 * @}
 */

/* /* end of group DMA_Register_Accessor_Macros */

```

```

/*
-----*
-- DMA Register Masks
----- */

```

```

/*!
 * @addtogroup DMA_Register_Masks DMA Register Masks
 * @{
 */


```

```

/* SAR Bit Fields */

#define DMA_SAR_SAR_MASK           0xFFFFFFFFu
#define DMA_SAR_SAR_SHIFT          0
#define DMA_SAR_SAR_WIDTH          32

```

```

#define DMA_SAR_SAR(x)
(((uint32_t)((uint32_t)(x)<<DMA_SAR_SAR_SHIFT))&DMA_SAR_SAR_MASK)

/* DAR Bit Fields */

#define DMA_DAR_DAR_MASK          0xFFFFFFFFu
#define DMA_DAR_DAR_SHIFT         0
#define DMA_DAR_DAR_WIDTH         32

#define DMA_DAR_DAR(x)
(((uint32_t)((uint32_t)(x)<<DMA_DAR_DAR_SHIFT))&DMA_DAR_DAR_MASK)

/* DSR_BCR Bit Fields */

#define DMA_DSR_BCR_BCR_MASK      0xFFFFFFFFu
#define DMA_DSR_BCR_BCR_SHIFT     0
#define DMA_DSR_BCR_BCR_WIDTH     24

#define DMA_DSR_BCR_BCR(x)
(((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BCR_SHIFT))&DMA_DSR_BCR_BCR_MASK)

#define DMA_DSR_BCR_DONE_MASK     0x1000000u
#define DMA_DSR_BCR_DONE_SHIFT    24
#define DMA_DSR_BCR_DONE_WIDTH    1

#define DMA_DSR_BCR_DONE(x)
(((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_DONE_SHIFT))&DMA_DSR_BCR_DONE_MASK)

#define DMA_DSR_BCR_BSY_MASK      0x2000000u
#define DMA_DSR_BCR_BSY_SHIFT     25
#define DMA_DSR_BCR_BSY_WIDTH     1

#define DMA_DSR_BCR_BSY(x)
(((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BSY_SHIFT))&DMA_DSR_BCR_BSY_MASK)

#define DMA_DSR_BCR_REQ_MASK      0x4000000u
#define DMA_DSR_BCR_REQ_SHIFT     26
#define DMA_DSR_BCR_REQ_WIDTH     1

#define DMA_DSR_BCR_REQ(x)
(((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_REQ_SHIFT))&DMA_DSR_BCR_REQ_MASK)

#define DMA_DSR_BCR_BED_MASK      0x10000000u
#define DMA_DSR_BCR_BED_SHIFT     28

```

```

#define DMA_DSR_BCR_BED_WIDTH          1
#define DMA_DSR_BCR_BED(x)             (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BED_SHIFT))&DMA_DSR_BCR_BED_MASK)
#define DMA_DSR_BCR_BES_MASK          0x20000000u
#define DMA_DSR_BCR_BES_SHIFT         29
#define DMA_DSR_BCR_BES_WIDTH         1
#define DMA_DSR_BCR_BES(x)             (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BES_SHIFT))&DMA_DSR_BCR_BES_MASK)
#define DMA_DSR_BCR_CE_MASK           0x40000000u
#define DMA_DSR_BCR_CE_SHIFT          30
#define DMA_DSR_BCR_CE_WIDTH          1
#define DMA_DSR_BCR_CE(x)              (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_CE_SHIFT))&DMA_DSR_BCR_CE_MASK)

/* DCR Bit Fields */

#define DMA_DCR_LCH2_MASK             0x3u
#define DMA_DCR_LCH2_SHIFT            0
#define DMA_DCR_LCH2_WIDTH            2
#define DMA_DCR_LCH2(x)               (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH2_SHIFT))&DMA_DCR_LCH2_MASK)
#define DMA_DCR_LCH1_MASK             0xCu
#define DMA_DCR_LCH1_SHIFT            2
#define DMA_DCR_LCH1_WIDTH            2
#define DMA_DCR_LCH1(x)               (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH1_SHIFT))&DMA_DCR_LCH1_MASK)
#define DMA_DCR_LINKCC_MASK           0x30u
#define DMA_DCR_LINKCC_SHIFT          4
#define DMA_DCR_LINKCC_WIDTH          2
#define DMA_DCR_LINKCC(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_LINKCC_SHIFT))&DMA_DCR_LINKCC_MASK)
#define DMA_DCR_D_REQ_MASK            0x80u
#define DMA_DCR_D_REQ_SHIFT           7

```

```

#define DMA_DCR_D_REQ_WIDTH          1
#define DMA_DCR_D_REQ(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_D_REQ_SHIFT))&DMA_DCR_D_REQ_MASK)
#define DMA_DCR_DMOD_MASK           0xF00u
#define DMA_DCR_DMOD_SHIFT          8
#define DMA_DCR_DMOD_WIDTH          4
#define DMA_DCR_DMOD(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DMOD_SHIFT))&DMA_DCR_DMOD_MASK)
#define DMA_DCR_SMOD_MASK           0xF000u
#define DMA_DCR_SMOD_SHIFT          12
#define DMA_DCR_SMOD_WIDTH          4
#define DMA_DCR_SMOD(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_SMOD_SHIFT))&DMA_DCR_SMOD_MASK)
#define DMA_DCR_START_MASK           0x10000u
#define DMA_DCR_START_SHIFT          16
#define DMA_DCR_START_WIDTH          1
#define DMA_DCR_START(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_START_SHIFT))&DMA_DCR_START_MASK)
#define DMA_DCR_DSIZEMASK           0x60000u
#define DMA_DCR_DSIZE_SHIFT          17
#define DMA_DCR_DSIZE_WIDTH          2
#define DMA_DCR_DSIZE(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DSIZE_SHIFT))&DMA_DCR_DSIZE_MASK)
#define DMA_DCR_DINC_MASK           0x80000u
#define DMA_DCR_DINC_SHIFT          19
#define DMA_DCR_DINC_WIDTH          1
#define DMA_DCR_DINC(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DINC_SHIFT))&DMA_DCR_DINC_MASK)
#define DMA_DCR_SSIZEMASK           0x300000u
#define DMA_DCR_SSIZE_SHIFT          20
#define DMA_DCR_SSIZE_WIDTH          2

```

```

#define DMA_DCR_SSIZE(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_SSIZE_SHIFT))&DMA_DCR_SSIZE_MASK)

#define DMA_DCR_SINC_MASK          0x400000u
#define DMA_DCR_SINC_SHIFT        22
#define DMA_DCR_SINC_WIDTH        1
#define DMA_DCR_SINC(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_SINC_SHIFT))&DMA_DCR_SINC_MASK)

#define DMA_DCR_EADREQ_MASK        0x800000u
#define DMA_DCR_EADREQ_SHIFT      23
#define DMA_DCR_EADREQ_WIDTH      1
#define DMA_DCR_EADREQ(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_EADREQ_SHIFT))&DMA_DCR_EADREQ_MASK)

#define DMA_DCR_AA_MASK           0x10000000u
#define DMA_DCR_AA_SHIFT          28
#define DMA_DCR_AA_WIDTH          1
#define DMA_DCR_AA(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_AA_SHIFT))&DMA_DCR_AA_MASK)

#define DMA_DCR_CS_MASK           0x20000000u
#define DMA_DCR_CS_SHIFT          29
#define DMA_DCR_CS_WIDTH          1
#define DMA_DCR_CS(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_CS_SHIFT))&DMA_DCR_CS_MASK)

#define DMA_DCR_ERQ_MASK          0x40000000u
#define DMA_DCR_ERQ_SHIFT         30
#define DMA_DCR_ERQ_WIDTH         1
#define DMA_DCR_ERQ(x)
(((uint32_t)((uint32_t)(x)<<DMA_DCR_ERQ_SHIFT))&DMA_DCR_ERQ_MASK)

#define DMA_DCR_EINT_MASK          0x80000000u
#define DMA_DCR_EINT_SHIFT         31
#define DMA_DCR_EINT_WIDTH         1

```

```
#define DMA_DCR_EINT(x)
(((uint32_t)((uint32_t)(x))<<DMA_DCR_EINT_SHIFT))&DMA_DCR_EINT_MASK)
```

```
/*!
 * @}
 */ /* end of group DMA_Register_Masks */
```

```
/* DMA - Peripheral instance base addresses */
/** Peripheral DMA base address */
#define DMA_BASE          (0x40008000u)
/** Peripheral DMA base pointer */
#define DMA0              ((DMA_Type *)DMA_BASE)
#define DMA_BASE_PTR      (DMA0)
/** Array initializer of DMA peripheral base addresses */
#define DMA_BASE_ADDRS    { DMA_BASE }
```

```
/** Array initializer of DMA peripheral base pointers */
#define DMA_BASE_PTRS     { DMA0 }
```

```
/* -----
-- DMA - Register accessor macros
----- */
```

```
/*!
 * @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros
 * @{
 */

```

```

/* DMA - Register instance definitions */

/* DMA */

#define DMA_SAR0           DMA_SAR_REG(DMA0,0)
#define DMA_DAR0           DMA_DAR_REG(DMA0,0)
#define DMA_DSR_BCR0        DMA_DSR_BCR_REG(DMA0,0)
#define DMA_DSR0           DMA_DSR_REG(DMA0,0)
#define DMA_DCR0           DMA_DCR_REG(DMA0,0)
#define DMA_SAR1           DMA_SAR_REG(DMA0,1)
#define DMA_DAR1           DMA_DAR_REG(DMA0,1)
#define DMA_DSR_BCR1        DMA_DSR_BCR_REG(DMA0,1)
#define DMA_DSR1           DMA_DSR_REG(DMA0,1)
#define DMA_DCR1           DMA_DCR_REG(DMA0,1)
#define DMA_SAR2           DMA_SAR_REG(DMA0,2)
#define DMA_DAR2           DMA_DAR_REG(DMA0,2)
#define DMA_DSR_BCR2        DMA_DSR_BCR_REG(DMA0,2)
#define DMA_DSR2           DMA_DSR_REG(DMA0,2)
#define DMA_DCR2           DMA_DCR_REG(DMA0,2)
#define DMA_SAR3           DMA_SAR_REG(DMA0,3)
#define DMA_DAR3           DMA_DAR_REG(DMA0,3)
#define DMA_DSR_BCR3        DMA_DSR_BCR_REG(DMA0,3)
#define DMA_DSR3           DMA_DSR_REG(DMA0,3)
#define DMA_DCR3           DMA_DCR_REG(DMA0,3)

```

```

/* DMA - Register array accessors */

#define DMA_SAR(index)      DMA_SAR_REG(DMA0,index)
#define DMA_DAR(index)      DMA_DAR_REG(DMA0,index)
#define DMA_DSR_BCR(index)   DMA_DSR_BCR_REG(DMA0,index)
#define DMA_DSR(index)      DMA_DSR_REG(DMA0,index)
#define DMA_DCR(index)      DMA_DCR_REG(DMA0,index)

```

```
/*!  
 * @}  
 */ /* end of group DMA_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group DMA_Peripheral_Access_Layer */
```

```
/* -----  
-- DMAMUX Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup DMAMUX_Peripheral_Access_Layer DMAMUX Peripheral Access Layer  
 * @{  
 */
```

```
/** DMAMUX - Register Layout Typedef */  
  
typedef struct {  
    __IO uint8_t CHCFG[4];           /**< Channel Configuration register, array offset: 0x0, array step:  
    0x1 */  
  
} DMAMUX_Type, *DMAMUX_MemMapPtr;
```

```
/* -----  
-- DMAMUX - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros  
 * @{  
 */
```

```
/* DMAMUX - Register accessors */  
  
#define DMAMUX_CHCFG_REG(base,index) ((base)->CHCFG[index])  
  
#define DMAMUX_CHCFG_COUNT 4
```

```
/*!  
 * @}  
 */ /* end of group DMAMUX_Register_Accessor_Macros */
```

```
/* -----  
-- DMAMUX Register Masks  
----- */
```

```
/*!  
 * @addtogroup DMAMUX_Register_Masks DMAMUX Register Masks  
 * @{  
 */
```

```
/* CHCFG Bit Fields */  
  
#define DMAMUX_CHCFG_SOURCE_MASK 0x3Fu  
  
#define DMAMUX_CHCFG_SOURCE_SHIFT 0  
  
#define DMAMUX_CHCFG_SOURCE_WIDTH 6
```

```

#define DMAMUX_CHCFG_SOURCE(x)
(((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_SOURCE_SHIFT))&DMAMUX_CHCFG_SOURCE_MASK

#define DMAMUX_CHCFG_TRIG_MASK          0x40u
#define DMAMUX_CHCFG_TRIG_SHIFT        6
#define DMAMUX_CHCFG_TRIG_WIDTH        1

#define DMAMUX_CHCFG_TRIG(x)
(((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_TRIG_SHIFT))&DMAMUX_CHCFG_TRIG_MASK

#define DMAMUX_CHCFG_ENBL_MASK         0x80u
#define DMAMUX_CHCFG_ENBL_SHIFT       7
#define DMAMUX_CHCFG_ENBL_WIDTH       1

#define DMAMUX_CHCFG_ENBL(x)
(((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_ENBL_SHIFT))&DMAMUX_CHCFG_ENBL_MASK

/*!
 * @}
 */

/* /* end of group DMAMUX_Register_Masks */

```

```

/* DMAMUX - Peripheral instance base addresses */

/** Peripheral DMAMUX0 base address */

#define DMAMUX0_BASE          (0x40021000u)

/** Peripheral DMAMUX0 base pointer */

#define DMAMUX0          ((DMAMUX_Type *)DMAMUX0_BASE)

#define DMAMUX0_BASE_PTR      (DMAMUX0)

/** Array initializer of DMAMUX peripheral base addresses */

#define DMAMUX_BASE_ADDRS     { DMAMUX0_BASE }

/** Array initializer of DMAMUX peripheral base pointers */

#define DMAMUX_BASE_PTRS      { DMAMUX0 }

/*
-----
```

```

-- DMAMUX - Register accessor macros
----- */
/*!
* @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros
* @{
*/
/* DMAMUX - Register instance definitions */
/* DMAMUX0 */
#define DMAMUX0_CHCFG0           DMAMUX_CHCFG_REG(DMAMUX0,0)
#define DMAMUX0_CHCFG1           DMAMUX_CHCFG_REG(DMAMUX0,1)
#define DMAMUX0_CHCFG2           DMAMUX_CHCFG_REG(DMAMUX0,2)
#define DMAMUX0_CHCFG3           DMAMUX_CHCFG_REG(DMAMUX0,3)

/* DMAMUX - Register array accessors */
#define DMAMUX0_CHCFG(index)      DMAMUX_CHCFG_REG(DMAMUX0,index)

/*!
* @}
*/
/* /* end of group DMAMUX_Register_Accessor_Macros */

/*!
* @}
*/
/* /* end of group DMAMUX_Peripheral_Access_Layer */

```

```

/*
-----  

-- FGPIO Peripheral Access Layer  

----- */
  

/*!  

* @addtogroup FGPIO_Peripheral_Access_Layer FGPIO Peripheral Access Layer
* @{
* /
  

/** FGPIO - Register Layout Typedef */  

typedef struct {  

    __IO uint32_t PDOR;           /**< Port Data Output Register, offset: 0x0 */  

    __O  uint32_t PSOR;          /**< Port Set Output Register, offset: 0x4 */  

    __O  uint32_t PCOR;          /**< Port Clear Output Register, offset: 0x8 */  

    __O  uint32_t PTOR;          /**< Port Toggle Output Register, offset: 0xC */  

    __I  uint32_t PDIR;          /**< Port Data Input Register, offset: 0x10 */  

    __IO uint32_t PDDR;          /**< Port Data Direction Register, offset: 0x14 */  

} FGPIO_Type, *FGPIO_MemMapPtr;  

  

/*
-----  

-- FGPIO - Register accessor macros  

----- */
  

/*!  

* @addtogroup FGPIO_Register_Accessor_Macros FGPIO - Register accessor macros
* @{
* /

```

```

/* FGPIO - Register accessors */

#define FGPIO_PDOR_REG(base)      ((base)->PDOR)
#define FGPIO_PSOR_REG(base)      ((base)->PSOR)
#define FGPIO_PCOR_REG(base)      ((base)->PCOR)
#define FGPIO_PTOR_REG(base)      ((base)->PTOR)
#define FGPIO_PDIR_REG(base)      ((base)->PDIR)
#define FGPIO_PDDR_REG(base)      ((base)->PDDR)

/*
 * @}
 */
/* /* end of group FGPIO_Register_Accessor_Macros */

```

```

/*
-----  

-- FGPIO Register Masks  

----- */

```

```

/*
* @addtogroup FGPIO_Register_Masks FGPIO Register Masks
* @{
*/

```

```

/* PDOR Bit Fields */

#define FGPIO_PDOR PDO_MASK      0xFFFFFFFFu
#define FGPIO_PDOR PDO_SHIFT     0
#define FGPIO_PDOR PDO_WIDTH    32

#define FGPIO_PDOR PDO(x)        (((uint32_t)(x))<<FGPIO_PDOR PDO_SHIFT)&FGPIO_PDOR PDO_MASK)

/* PSOR Bit Fields */

```

```

#define GPIO_PSOR_PTSO_MASK      0xFFFFFFFFu
#define GPIO_PSOR_PTSO_SHIFT     0
#define GPIO_PSOR_PTSO_WIDTH     32
#define GPIO_PSOR_PTSO(x)        (((uint32_t)((uint32_t)(x)<<GPIO_PSOR_PTSO_SHIFT))&GPIO_PSOR_PTSO_MASK)

/* PCOR Bit Fields */

#define GPIO_PCOR_PTCO_MASK      0xFFFFFFFFu
#define GPIO_PCOR_PTCO_SHIFT     0
#define GPIO_PCOR_PTCO_WIDTH     32
#define GPIO_PCOR_PTCO(x)        (((uint32_t)((uint32_t)(x)<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)

/* PTOR Bit Fields */

#define GPIO_PTOR_PTTO_MASK      0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT     0
#define GPIO_PTOR_PTTO_WIDTH     32
#define GPIO_PTOR_PTTO(x)        (((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/* PDIR Bit Fields */

#define GPIO_PDIR_PDI_MASK       0xFFFFFFFFu
#define GPIO_PDIR_PDI_SHIFT      0
#define GPIO_PDIR_PDI_WIDTH      32
#define GPIO_PDIR_PDI(x)         (((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK       0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT      0
#define GPIO_PDDR_PDD_WIDTH      32
#define GPIO_PDDR_PDD(x)         (((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

```

```

/*!
* @}

/* /* end of group FGPIO_Register_Masks */

/* FGPIO - Peripheral instance base addresses */

/** Peripheral FGPIOA base address */

#define FGPIOA_BASE          (0xF80FF000u)

/** Peripheral FGPIOA base pointer */

#define FGPIOA              ((FGPIO_Type *)FGPIOA_BASE)

#define FGPIOA_BASE_PTR      (FGPIOA)

/** Peripheral FGPIOB base address */

#define FGPIOB_BASE          (0xF80FF040u)

/** Peripheral FGPIOB base pointer */

#define FGPIOB              ((FGPIO_Type *)FGPIOB_BASE)

#define FGPIOB_BASE_PTR      (FGPIOB)

/** Peripheral FGPIOC base address */

#define FGPIOC_BASE          (0xF80FF080u)

/** Peripheral FGPIOC base pointer */

#define FGPIOC              ((FGPIO_Type *)FGPIOC_BASE)

#define FGPIOC_BASE_PTR      (FGPIOC)

/** Peripheral FGPIOD base address */

#define FGPIOD_BASE          (0xF80FF0C0u)

/** Peripheral FGPIOD base pointer */

#define FGPIOD              ((FGPIO_Type *)FGPIOD_BASE)

#define FGPIOD_BASE_PTR      (FGPIOD)

/** Peripheral FGPIOE base address */

#define FGPIOE_BASE          (0xF80FF100u)

/** Peripheral FGPIOE base pointer */

```

```

#define FGPIOE ((FGPIO_Type *)FGPIOE_BASE)
#define FGPIOE_BASE_PTR (FGPIOE)
/** Array initializer of GPIO peripheral base addresses */
#define FGPIO_BASE_ADDRS { FGPIOA_BASE, FGPIOB_BASE, FGPIOC_BASE, FGPIOD_BASE,
FGPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */
#define FGPIO_BASE_PTRS { FGPIOA, FGPIOB, FGPIOC, FGPIOD, FGPIOE }

```

```

/*
-- GPIO - Register accessor macros
----- */

```

```

/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{
*/

```

```

/* GPIO - Register instance definitions */

/* GPIOA */

#define FGPIOA_PDOR      GPIO_PDOR_REG(GPIOA)
#define FGPIOA_PSOR      GPIO_PSOR_REG(GPIOA)
#define FGPIOA_PCOR      GPIO_PCOR_REG(GPIOA)
#define FGPIOA_PTOR      GPIO_PTOR_REG(GPIOA)
#define FGPIOA_PDIR      GPIO_PDIR_REG(GPIOA)
#define FGPIOA_PDDR      GPIO_PDDR_REG(GPIOA)

/* GPIOB */

#define FGPIOB_PDOR      GPIO_PDOR_REG(GPIOB)
#define FGPIOB_PSOR      GPIO_PSOR_REG(GPIOB)

```

```

#define FGPIOB_PCOR           FGPIO_PCOR_REG(FGPIOB)
#define FGPIOB_PTOR           FGPIO_PTOR_REG(FGPIOB)
#define FGPIOB_PDIR           FGPIO_PDIR_REG(FGPIOB)
#define FGPIOB_PDDR           FGPIO_PDDR_REG(FGPIOB)

/* FGPIOC */

#define FGPIOC_PDOR           FGPIO_PDOR_REG(FGPIOC)
#define FGPIOC_PSOR           FGPIO_PSOR_REG(FGPIOC)
#define FGPIOC_PCOR           FGPIO_PCOR_REG(FGPIOC)
#define FGPIOC_PTOR           FGPIO_PTOR_REG(FGPIOC)
#define FGPIOC_PDIR           FGPIO_PDIR_REG(FGPIOC)
#define FGPIOC_PDDR           FGPIO_PDDR_REG(FGPIOC)

/* FGPIOD */

#define FGPIOD_PDOR           FGPIO_PDOR_REG(FGPIOD)
#define FGPIOD_PSOR           FGPIO_PSOR_REG(FGPIOD)
#define FGPIOD_PCOR           FGPIO_PCOR_REG(FGPIOD)
#define FGPIOD_PTOR           FGPIO_PTOR_REG(FGPIOD)
#define FGPIOD_PDIR           FGPIO_PDIR_REG(FGPIOD)
#define FGPIOD_PDDR           FGPIO_PDDR_REG(FGPIOD)

/* FGPIOE */

#define FGPIOE_PDOR           FGPIO_PDOR_REG(FGPIOE)
#define FGPIOE_PSOR           FGPIO_PSOR_REG(FGPIOE)
#define FGPIOE_PCOR           FGPIO_PCOR_REG(FGPIOE)
#define FGPIOE_PTOR           FGPIO_PTOR_REG(FGPIOE)
#define FGPIOE_PDIR           FGPIO_PDIR_REG(FGPIOE)
#define FGPIOE_PDDR           FGPIO_PDDR_REG(FGPIOE)

/*
 * @}
 */

/* /* end of group GPIO_Register_Accessor_Macros */

```

```

/*!
* @}
/* /* end of group FGPIO_Peripheral_Access_Layer */

/*
-----  

-- FTFA Peripheral Access Layer  

----- */
/*!  

* @addtogroup FTFA_Peripheral_Access_Layer FTFA Peripheral Access Layer
* @{
*/
  

/** FTFA - Register Layout Typedef */
typedef struct {
    __IO uint8_t FSTAT;           /**< Flash Status Register, offset: 0x0 */
    __IO uint8_t FCNFG;          /**< Flash Configuration Register, offset: 0x1 */
    __I  uint8_t FSEC;           /**< Flash Security Register, offset: 0x2 */
    __I  uint8_t FOPT;           /**< Flash Option Register, offset: 0x3 */
    __IO uint8_t FCCOB3;         /**< Flash Common Command Object Registers, offset: 0x4 */
    __IO uint8_t FCCOB2;         /**< Flash Common Command Object Registers, offset: 0x5 */
    __IO uint8_t FCCOB1;         /**< Flash Common Command Object Registers, offset: 0x6 */
    __IO uint8_t FCCOBO;         /**< Flash Common Command Object Registers, offset: 0x7 */
    __IO uint8_t FCCOB7;         /**< Flash Common Command Object Registers, offset: 0x8 */
    __IO uint8_t FCCOB6;         /**< Flash Common Command Object Registers, offset: 0x9 */
    __IO uint8_t FCCOB5;         /**< Flash Common Command Object Registers, offset: 0xA */
}

```

```

__IO uint8_t FCCOB4;           /**< Flash Common Command Object Registers, offset: 0xB */
__IO uint8_t FCCOBB;          /**< Flash Common Command Object Registers, offset: 0xC */
__IO uint8_t FCCOBA;          /**< Flash Common Command Object Registers, offset: 0xD */
__IO uint8_t FCCOB9;          /**< Flash Common Command Object Registers, offset: 0xE */
__IO uint8_t FCCOB8;          /**< Flash Common Command Object Registers, offset: 0xF */
__IO uint8_t FPROT3;          /**< Program Flash Protection Registers, offset: 0x10 */
__IO uint8_t FPROT2;          /**< Program Flash Protection Registers, offset: 0x11 */
__IO uint8_t FPROT1;          /**< Program Flash Protection Registers, offset: 0x12 */
__IO uint8_t FPROT0;          /**< Program Flash Protection Registers, offset: 0x13 */

} FTFA_Type, *FTFA_MemMapPtr;

```

```

/*
-----  

-- FTFA - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros  

* @{
* /  


```

```

/* FTFA - Register accessors */  

#define FTFA_FSTAT_REG(base)      ((base)->FSTAT)  

#define FTFA_FCNFG_REG(base)      ((base)->FCNFG)  

#define FTFA_FSEC_REG(base)       ((base)->FSEC)  

#define FTFA_FOPT_REG(base)       ((base)->FOPT)  

#define FTFA_FCCOB3_REG(base)     ((base)->FCCOB3)  

#define FTFA_FCCOB2_REG(base)     ((base)->FCCOB2)  

#define FTFA_FCCOB1_REG(base)     ((base)->FCCOB1)

```

```
#define FTFA_FCCOBO_REG(base) ((base)->FCCOBO)
#define FTFA_FCCOB7_REG(base) ((base)->FCCOB7)
#define FTFA_FCCOB6_REG(base) ((base)->FCCOB6)
#define FTFA_FCCOB5_REG(base) ((base)->FCCOB5)
#define FTFA_FCCOB4_REG(base) ((base)->FCCOB4)
#define FTFA_FCCOBB_REG(base) ((base)->FCCOBB)
#define FTFA_FCCOBA_REG(base) ((base)->FCCOBA)
#define FTFA_FCCOB9_REG(base) ((base)->FCCOB9)
#define FTFA_FCCOB8_REG(base) ((base)->FCCOB8)
#define FTFA_FPROT3_REG(base) ((base)->FPROT3)
#define FTFA_FPROT2_REG(base) ((base)->FPROT2)
#define FTFA_FPROT1_REG(base) ((base)->FPROT1)
#define FTFA_FPROTO_REG(base) ((base)->FPROTO)
```

```
/*
 * @}
 */
/* /* end of group FTFA_Register_Accessor_Macros */
```

```
/*
-----  
-- FTFA Register Masks  
----- */
```

```
/*
 * @addtogroup FTFA_Register_Masks FTFA Register Masks
 * @{
 */

```

```
/* FSTAT Bit Fields */
```

```

#define FTFA_FSTAT_MGSTAT0_MASK          0x1u
#define FTFA_FSTAT_MGSTAT0_SHIFT         0
#define FTFA_FSTAT_MGSTAT0_WIDTH         1
#define FTFA_FSTAT_MGSTAT0(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_MGSTAT0_SHIFT))&FTFA_FSTAT_MGSTAT0_MASK)

#define FTFA_FSTAT_FPVIOL_MASK          0x10u
#define FTFA_FSTAT_FPVIOL_SHIFT         4
#define FTFA_FSTAT_FPVIOL_WIDTH         1
#define FTFA_FSTAT_FPVIOL(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_FPVIOL_SHIFT))&FTFA_FSTAT_FPVIOL_MASK)

#define FTFA_FSTAT_ACCERR_MASK          0x20u
#define FTFA_FSTAT_ACCERR_SHIFT         5
#define FTFA_FSTAT_ACCERR_WIDTH         1
#define FTFA_FSTAT_ACCERR(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_ACCERR_SHIFT))&FTFA_FSTAT_ACCERR_MASK)

#define FTFA_FSTAT_RDCOLERR_MASK         0x40u
#define FTFA_FSTAT_RDCOLERR_SHIFT        6
#define FTFA_FSTAT_RDCOLERR_WIDTH        1
#define FTFA_FSTAT_RDCOLERR(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_RDCOLERR_SHIFT))&FTFA_FSTAT_RDCOLERR_MASK)

#define FTFA_FSTAT_CCIF_MASK            0x80u
#define FTFA_FSTAT_CCIF_SHIFT           7
#define FTFA_FSTAT_CCIF_WIDTH           1
#define FTFA_FSTAT_CCIF(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_CCIF_SHIFT))&FTFA_FSTAT_CCIF_MASK)

/* FCNFG Bit Fields */

#define FTFA_FCNFG_ERSSUSP_MASK          0x10u
#define FTFA_FCNFG_ERSSUSP_SHIFT         4
#define FTFA_FCNFG_ERSSUSP_WIDTH         1
#define FTFA_FCNFG_ERSSUSP(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_ERSSUSP_SHIFT))&FTFA_FCNFG_ERSSUSP_MASK)

```

```

#define FTFA_FCNFG_ERSAREQ_MASK          0x20u
#define FTFA_FCNFG_ERSAREQ_SHIFT         5
#define FTFA_FCNFG_ERSAREQ_WIDTH         1
#define FTFA_FCNFG_ERSAREQ(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_ERSAREQ_SHIFT))&FTFA_FCNFG_ERSAREQ_MASK)

#define FTFA_FCNFG_RDCOLLIE_MASK         0x40u
#define FTFA_FCNFG_RDCOLLIE_SHIFT        6
#define FTFA_FCNFG_RDCOLLIE_WIDTH        1
#define FTFA_FCNFG_RDCOLLIE(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_RDCOLLIE_SHIFT))&FTFA_FCNFG_RDCOLLIE_MASK)

#define FTFA_FCNFG_CCIE_MASK             0x80u
#define FTFA_FCNFG_CCIE_SHIFT            7
#define FTFA_FCNFG_CCIE_WIDTH            1
#define FTFA_FCNFG_CCIE(x)              (((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_CCIE_SHIFT))&FTFA_FCNFG_CCIE_MASK)

/* FSEC Bit Fields */

#define FTFA_FSEC_SEC_MASK               0x3u
#define FTFA_FSEC_SEC_SHIFT              0
#define FTFA_FSEC_SEC_WIDTH              2
#define FTFA_FSEC_SEC(x)                (((uint8_t)((uint8_t)(x))<<FTFA_FSEC_SEC_SHIFT))&FTFA_FSEC_SEC_MASK)

#define FTFA_FSEC_FSLACC_MASK            0xCu
#define FTFA_FSEC_FSLACC_SHIFT           2
#define FTFA_FSEC_FSLACC_WIDTH           2
#define FTFA_FSEC_FSLACC(x)              (((uint8_t)((uint8_t)(x))<<FTFA_FSEC_FSLACC_SHIFT))&FTFA_FSEC_FSLACC_MASK)

#define FTFA_FSEC_MEEN_MASK              0x30u
#define FTFA_FSEC_MEEN_SHIFT             4
#define FTFA_FSEC_MEEN_WIDTH              2
#define FTFA_FSEC_MEEN(x)                (((uint8_t)((uint8_t)(x))<<FTFA_FSEC_MEEN_SHIFT))&FTFA_FSEC_MEEN_MASK)

```

```

#define FTFA_FSEC_KEYEN_MASK          0xCOu
#define FTFA_FSEC_KEYEN_SHIFT         6
#define FTFA_FSEC_KEYEN_WIDTH         2
#define FTFA_FSEC_KEYEN(x) (((uint8_t)((uint8_t)(x))<<FTFA_FSEC_KEYEN_SHIFT))&FTFA_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define FTFA_FOPT_OPT_MASK           0xFFu
#define FTFA_FOPT_OPT_SHIFT          0
#define FTFA_FOPT_OPT_WIDTH          8
#define FTFA_FOPT_OPT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FOPT_OPT_SHIFT))&FTFA_FOPT_OPT_MASK)

/* FCCOB3 Bit Fields */

#define FTFA_FCCOB3_CCOBn_MASK        0xFFu
#define FTFA_FCCOB3_CCOBn_SHIFT       0
#define FTFA_FCCOB3_CCOBn_WIDTH       8
#define FTFA_FCCOB3_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB3_CCOBn_SHIFT))&FTFA_FCCOB3_CCOBn_MASK)

/* FCCOB2 Bit Fields */

#define FTFA_FCCOB2_CCOBn_MASK        0xFFu
#define FTFA_FCCOB2_CCOBn_SHIFT       0
#define FTFA_FCCOB2_CCOBn_WIDTH       8
#define FTFA_FCCOB2_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB2_CCOBn_SHIFT))&FTFA_FCCOB2_CCOBn_MASK)

/* FCCOB1 Bit Fields */

#define FTFA_FCCOB1_CCOBn_MASK        0xFFu
#define FTFA_FCCOB1_CCOBn_SHIFT       0
#define FTFA_FCCOB1_CCOBn_WIDTH       8
#define FTFA_FCCOB1_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB1_CCOBn_SHIFT))&FTFA_FCCOB1_CCOBn_MASK)

/* FCCOB0 Bit Fields */

```

```

#define FTFA_FCCOB0_CCOBn_MASK      0xFFu
#define FTFA_FCCOB0_CCOBn_SHIFT     0
#define FTFA_FCCOB0_CCOBn_WIDTH     8
#define FTFA_FCCOB0_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB0_CCOBn_SHIFT))&FTFA_FCCOB0_CCOBn_MASK)

/* FCCOB7 Bit Fields */

#define FTFA_FCCOB7_CCOBn_MASK      0xFFu
#define FTFA_FCCOB7_CCOBn_SHIFT     0
#define FTFA_FCCOB7_CCOBn_WIDTH     8
#define FTFA_FCCOB7_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB7_CCOBn_SHIFT))&FTFA_FCCOB7_CCOBn_MASK)

/* FCCOB6 Bit Fields */

#define FTFA_FCCOB6_CCOBn_MASK      0xFFu
#define FTFA_FCCOB6_CCOBn_SHIFT     0
#define FTFA_FCCOB6_CCOBn_WIDTH     8
#define FTFA_FCCOB6_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB6_CCOBn_SHIFT))&FTFA_FCCOB6_CCOBn_MASK)

/* FCCOB5 Bit Fields */

#define FTFA_FCCOB5_CCOBn_MASK      0xFFu
#define FTFA_FCCOB5_CCOBn_SHIFT     0
#define FTFA_FCCOB5_CCOBn_WIDTH     8
#define FTFA_FCCOB5_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB5_CCOBn_SHIFT))&FTFA_FCCOB5_CCOBn_MASK)

/* FCCOB4 Bit Fields */

#define FTFA_FCCOB4_CCOBn_MASK      0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT     0
#define FTFA_FCCOB4_CCOBn_WIDTH     8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK)

/* FCCOBB Bit Fields */

```

```

#define FTFA_FCCOBB_CCOBn_MASK      0xFFu
#define FTFA_FCCOBB_CCOBn_SHIFT     0
#define FTFA_FCCOBB_CCOBn_WIDTH      8
#define FTFA_FCCOBB_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOBB_CCOBn_SHIFT))&FTFA_FCCOBB_CCOBn_MASK)

/* FCCOBA Bit Fields */

#define FTFA_FCCOBA_CCOBn_MASK      0xFFu
#define FTFA_FCCOBA_CCOBn_SHIFT     0
#define FTFA_FCCOBA_CCOBn_WIDTH      8
#define FTFA_FCCOBA_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOBA_CCOBn_SHIFT))&FTFA_FCCOBA_CCOBn_MASK)

/* FCCOB9 Bit Fields */

#define FTFA_FCCOB9_CCOBn_MASK      0xFFu
#define FTFA_FCCOB9_CCOBn_SHIFT     0
#define FTFA_FCCOB9_CCOBn_WIDTH      8
#define FTFA_FCCOB9_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB9_CCOBn_SHIFT))&FTFA_FCCOB9_CCOBn_MASK)

/* FCCOB8 Bit Fields */

#define FTFA_FCCOB8_CCOBn_MASK      0xFFu
#define FTFA_FCCOB8_CCOBn_SHIFT     0
#define FTFA_FCCOB8_CCOBn_WIDTH      8
#define FTFA_FCCOB8_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB8_CCOBn_SHIFT))&FTFA_FCCOB8_CCOBn_MASK)

/* FPROT3 Bit Fields */

#define FTFA_FPROT3_PROT_MASK       0xFFu
#define FTFA_FPROT3_PROT_SHIFT      0
#define FTFA_FPROT3_PROT_WIDTH      8
#define FTFA_FPROT3_PROT(x) (((uint8_t)((uint8_t)(x)<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

```

```

#define FTFA_FPROT2_PROT_MASK          0xFFu
#define FTFA_FPROT2_PROT_SHIFT         0
#define FTFA_FPROT2_PROT_WIDTH         8
#define FTFA_FPROT2_PROT(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FPROT2_PROT_SHIFT))&FTFA_FPROT2_PROT_MASK)

/* FPROT1 Bit Fields */

#define FTFA_FPROT1_PROT_MASK          0xFFu
#define FTFA_FPROT1_PROT_SHIFT         0
#define FTFA_FPROT1_PROT_WIDTH         8
#define FTFA_FPROT1_PROT(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FPROT1_PROT_SHIFT))&FTFA_FPROT1_PROT_MASK)

/* FPROT0 Bit Fields */

#define FTFA_FPROT0_PROT_MASK          0xFFu
#define FTFA_FPROT0_PROT_SHIFT         0
#define FTFA_FPROT0_PROT_WIDTH         8
#define FTFA_FPROT0_PROT(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FPROT0_PROT_SHIFT))&FTFA_FPROT0_PROT_MASK)

/*
 * @}
 */

/* /* end of group FTFA_Register_Masks */

```

```

/* FTFA - Peripheral instance base addresses */

/** Peripheral FTFA base address */

#define FTFA_BASE                  (0x40020000u)

/** Peripheral FTFA base pointer */

#define FTFA                      ((FTFA_Type *)FTFA_BASE)

#define FTFA_BASE_PTR              (FTFA)

/** Array initializer of FTFA peripheral base addresses */

```

```

#define FTFA_BASE_ADDRS          { FTFA_BASE }

/** Array initializer of FTFA peripheral base pointers */

#define FTFA_BASE_PTRS           { FTFA }

/*
-----  

-- FTFA - Register accessor macros  

----- */

```

```

/*!  

 * @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros  

 * @{
 * /
```

```

/* FTFA - Register instance definitions */

/* FTFA */

#define FTFA_FSTAT                FTFA_FSTAT_REG(FTFA)  

#define FTFA_FCNG                 FTFA_FCNG_REG(FTFA)  

#define FTFA_FSEC                 FTFA_FSEC_REG(FTFA)  

#define FTFA_FOPT                 FTFA_FOPT_REG(FTFA)  

#define FTFA_FCCOB3               FTFA_FCCOB3_REG(FTFA)  

#define FTFA_FCCOB2               FTFA_FCCOB2_REG(FTFA)  

#define FTFA_FCCOB1               FTFA_FCCOB1_REG(FTFA)  

#define FTFA_FCCOB0               FTFA_FCCOB0_REG(FTFA)  

#define FTFA_FCCOB7               FTFA_FCCOB7_REG(FTFA)  

#define FTFA_FCCOB6               FTFA_FCCOB6_REG(FTFA)  

#define FTFA_FCCOB5               FTFA_FCCOB5_REG(FTFA)  

#define FTFA_FCCOB4               FTFA_FCCOB4_REG(FTFA)  

#define FTFA_FCCOB8               FTFA_FCCOB8_REG(FTFA)
```

```
#define FTFA_FCCOBA           FTFA_FCCOBA_REG(FTFA)
#define FTFA_FCCOB9           FTFA_FCCOB9_REG(FTFA)
#define FTFA_FCCOB8           FTFA_FCCOB8_REG(FTFA)
#define FTFA_FPROT3           FTFA_FPROT3_REG(FTFA)
#define FTFA_FPROT2           FTFA_FPROT2_REG(FTFA)
#define FTFA_FPROT1           FTFA_FPROT1_REG(FTFA)
#define FTFA_FPROTO            FTFA_FPROTO_REG(FTFA)
```

```
/*
 * @}
 */
/* end of group FTFA_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group FTFA_Peripheral_Access_Layer */
```

```
/*
-----  
-- GPIO Peripheral Access Layer  
----- */
```

```
/*
 * @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access Layer
 * @{
 */

```

```
/** GPIO - Register Layout Typedef */
typedef struct {
```

```

__IO uint32_t PDOR;           /**< Port Data Output Register, offset: 0x0 */
__O uint32_t PSOR;           /**< Port Set Output Register, offset: 0x4 */
__O uint32_t PCOR;           /**< Port Clear Output Register, offset: 0x8 */
__O uint32_t PTOR;           /**< Port Toggle Output Register, offset: 0xC */
__I uint32_t PDIR;           /**< Port Data Input Register, offset: 0x10 */
__IO uint32_t PDDR;           /**< Port Data Direction Register, offset: 0x14 */

} GPIO_Type, *GPIO_MemMapPtr;

```

```

/*
-----*
-- GPIO - Register accessor macros
----- */

```

```

/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{
*/

```

```

/* GPIO - Register accessors */

#define GPIO_PDOR_REG(base)      ((base)->PDOR)
#define GPIO_PSOR_REG(base)      ((base)->PSOR)
#define GPIO_PCOR_REG(base)      ((base)->PCOR)
#define GPIO_PTOR_REG(base)      ((base)->PTOR)
#define GPIO_PDIR_REG(base)      ((base)->PDIR)
#define GPIO_PDDR_REG(base)      ((base)->PDDR)
```

```

/*!
* @}
*/
/* /* end of group GPIO_Register_Accessor_Macros */
```

```

/*
-----  

-- GPIO Register Masks  

----- */
  

/*!  

 * @addtogroup GPIO_Register_Masks GPIO Register Masks
 * @{
 * /  

  

/* PDOR Bit Fields */  

#define GPIO_PDOR_PDO_MASK          0xFFFFFFFFu  

#define GPIO_PDOR_PDO_SHIFT         0  

#define GPIO_PDOR_PDO_WIDTH         32  

#define GPIO_PDOR_PDO(x)            (((uint32_t)(x))<<GPIO_PDOR_PDO_SHIFT)&GPIO_PDOR_PDO_MASK)  

  

/* PSOR Bit Fields */  

#define GPIO_PSOR PTSO MASK        0xFFFFFFFFu  

#define GPIO_PSOR PTSO SHIFT       0  

#define GPIO_PSOR PTSO WIDTH       32  

#define GPIO_PSOR PTSO(x)          (((uint32_t)(x))<<GPIO_PSOR_PTSO_SHIFT)&GPIO_PSOR_PTSO_MASK)  

  

/* PCOR Bit Fields */  

#define GPIO_PCOR_PTCO MASK        0xFFFFFFFFu  

#define GPIO_PCOR_PTCO SHIFT       0  

#define GPIO_PCOR_PTCO WIDTH       32  

#define GPIO_PCOR_PTCO(x)          (((uint32_t)(x))<<GPIO_PCOR_PTCO_SHIFT)&GPIO_PCOR_PTCO_MASK)  

  

/* PTOR Bit Fields */

```

```

#define GPIO_PTOR_PTTO_MASK          0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT         0
#define GPIO_PTOR_PTTO_WIDTH         32
#define GPIO_PTOR_PTTO(x)            (((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/* PDIR Bit Fields */

#define GPIO_PDIR_PDI_MASK          0xFFFFFFFFu
#define GPIO_PDIR_PDI_SHIFT         0
#define GPIO_PDIR_PDI_WIDTH         32
#define GPIO_PDIR_PDI(x)             (((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK          0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT         0
#define GPIO_PDDR_PDD_WIDTH         32
#define GPIO_PDDR_PDD(x)             (((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

/*
 * @}
 */

/* /* end of group GPIO_Register_Masks */

```

```

/* GPIO - Peripheral instance base addresses */

/** Peripheral GPIOA base address */

#define GPIOA_BASE                  (0x400FF000u)

/** Peripheral GPIOA base pointer */

#define GPIOA                      ((GPIO_Type *)GPIOA_BASE)

#define GPIOA_BASE_PTR              (GPIOA)

/** Peripheral GPIOB base address */

```

```

#define GPIOB_BASE          (0x400FF040u)
/** Peripheral GPIOB base pointer */

#define GPIOB          ((GPIO_Type *)GPIOB_BASE)
#define GPIOB_BASE_PTR      (GPIOB)

/** Peripheral GPIOC base address */
#define GPIOC_BASE          (0x400FF080u)
/** Peripheral GPIOC base pointer */

#define GPIOC          ((GPIO_Type *)GPIOC_BASE)
#define GPIOC_BASE_PTR      (GPIOC)

/** Peripheral GPIOD base address */
#define GPIOD_BASE          (0x400FF0C0u)
/** Peripheral GPIOD base pointer */

#define GPIOD          ((GPIO_Type *)GPIOD_BASE)
#define GPIOD_BASE_PTR      (GPIOD)

/** Peripheral GPIOE base address */
#define GPIOE_BASE          (0x400FF100u)
/** Peripheral GPIOE base pointer */

#define GPIOE          ((GPIO_Type *)GPIOE_BASE)
#define GPIOE_BASE_PTR      (GPIOE)

/** Array initializer of GPIO peripheral base addresses */
#define GPIO_BASE_ADDRS      { GPIOA_BASE, GPIOB_BASE, GPIOC_BASE, GPIOD_BASE,
                           GPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */
#define GPIO_BASE_PTRS        { GPIOA, GPIOB, GPIOC, GPIOD, GPIOE }

/*
-----*
-- GPIO - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros  
 * @{  
 */
```

```
/* GPIO - Register instance definitions */  
  
/* GPIOA */  
  
#define GPIOA_PDOR          GPIO_PDOR_REG(GPIOA)  
#define GPIOA_PSOR          GPIO_PSOR_REG(GPIOA)  
#define GPIOA_PCOR          GPIO_PCOR_REG(GPIOA)  
#define GPIOA_PTOR          GPIO_PTOR_REG(GPIOA)  
#define GPIOA_PDIR          GPIO_PDIR_REG(GPIOA)  
#define GPIOA_PDDR          GPIO_PDDR_REG(GPIOA)  
  
/* GPIOB */  
  
#define GPIOB_PDOR          GPIO_PDOR_REG(GPIOB)  
#define GPIOB_PSOR          GPIO_PSOR_REG(GPIOB)  
#define GPIOB_PCOR          GPIO_PCOR_REG(GPIOB)  
#define GPIOB_PTOR          GPIO_PTOR_REG(GPIOB)  
#define GPIOB_PDIR          GPIO_PDIR_REG(GPIOB)  
#define GPIOB_PDDR          GPIO_PDDR_REG(GPIOB)  
  
/* GPIOC */  
  
#define GPIOC_PDOR          GPIO_PDOR_REG(GPIOC)  
#define GPIOC_PSOR          GPIO_PSOR_REG(GPIOC)  
#define GPIOC_PCOR          GPIO_PCOR_REG(GPIOC)  
#define GPIOC_PTOR          GPIO_PTOR_REG(GPIOC)  
#define GPIOC_PDIR          GPIO_PDIR_REG(GPIOC)  
#define GPIOC_PDDR          GPIO_PDDR_REG(GPIOC)  
  
/* GPIOD */
```

```
#define GPIOD_PDOR          GPIO_PDOR_REG(GPIOD)
#define GPIOD_PSOR          GPIO_PSOR_REG(GPIOD)
#define GPIOD_PCOR          GPIO_PCOR_REG(GPIOD)
#define GPIOD_PTOR          GPIO_PTOR_REG(GPIOD)
#define GPIOD_PDIR          GPIO_PDIR_REG(GPIOD)
#define GPIOD_PDDR          GPIO_PDDR_REG(GPIOD)

/* GPIOE */

#define GPIOE_PDOR          GPIO_PDOR_REG(GPIOE)
#define GPIOE_PSOR          GPIO_PSOR_REG(GPIOE)
#define GPIOE_PCOR          GPIO_PCOR_REG(GPIOE)
#define GPIOE_PTOR          GPIO_PTOR_REG(GPIOE)
#define GPIOE_PDIR          GPIO_PDIR_REG(GPIOE)
#define GPIOE_PDDR          GPIO_PDDR_REG(GPIOE)

/*!

 * @}

 */ /* end of group GPIO_Register_Accessor_Macros */
```

```
/*!

 * @}

 */ /* end of group GPIO_Peripheral_Access_Layer */
```

```
/*
-----  
-- I2C Peripheral Access Layer  
----- */
```

```
/*!
```

```

* @addtogroup I2C_Peripheral_Access_Layer I2C Peripheral Access Layer

* @{
* /


/** I2C - Register Layout Typedef */

typedef struct {

    __IO uint8_t A1;           /**< I2C Address Register 1, offset: 0x0 */

    __IO uint8_t F;            /**< I2C Frequency Divider register, offset: 0x1 */

    __IO uint8_t C1;           /**< I2C Control Register 1, offset: 0x2 */

    __IO uint8_t S;            /**< I2C Status register, offset: 0x3 */

    __IO uint8_t D;            /**< I2C Data I/O register, offset: 0x4 */

    __IO uint8_t C2;           /**< I2C Control Register 2, offset: 0x5 */

    __IO uint8_t FLT;          /**< I2C Programmable Input Glitch Filter register, offset: 0x6 */

    __IO uint8_t RA;           /**< I2C Range Address register, offset: 0x7 */

    __IO uint8_t SMB;          /**< I2C SMBus Control and Status register, offset: 0x8 */

    __IO uint8_t A2;           /**< I2C Address Register 2, offset: 0x9 */

    __IO uint8_t SLTH;          /**< I2C SCL Low Timeout Register High, offset: 0xA */

    __IO uint8_t SLTL;          /**< I2C SCL Low Timeout Register Low, offset: 0xB */

} I2C_Type, *I2C_MemMapPtr;

/* -----
-- I2C - Register accessor macros
----- */

/*!

* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros
* @{
* /

```

```
/* I2C - Register accessors */

#define I2C_A1_REG(base)          ((base)->A1)
#define I2C_F_REG(base)           ((base)->F)
#define I2C_C1_REG(base)          ((base)->C1)
#define I2C_S_REG(base)           ((base)->S)
#define I2C_D_REG(base)           ((base)->D)
#define I2C_C2_REG(base)          ((base)->C2)
#define I2C_FLT_REG(base)         ((base)->FLT)
#define I2C_RA_REG(base)          ((base)->RA)
#define I2C_SMB_REG(base)         ((base)->SMB)
#define I2C_A2_REG(base)          ((base)->A2)
#define I2C_SLTH_REG(base)        ((base)->SLTH)
#define I2C_SLTL_REG(base)        ((base)->SLTL)
```

```
/*
 * @}
 */
/* /* end of group I2C_Register_Accessor_Macros */
```

```
/*
-----  
-- I2C Register Masks  
----- */
```

```
/*
* @addtogroup I2C_Register_Masks I2C Register Masks
* @{
*/

```

```

/* A1 Bit Fields */

#define I2C_A1_AD_MASK          0xFEu
#define I2C_A1_AD_SHIFT         1
#define I2C_A1_AD_WIDTH         7
#define I2C_A1_AD(x)            (((uint8_t)((uint8_t)(x)<<I2C_A1_AD_SHIFT))&I2C_A1_AD_MASK)

/* F Bit Fields */

#define I2C_F_ICR_MASK          0x3Fu
#define I2C_F_ICR_SHIFT         0
#define I2C_F_ICR_WIDTH         6
#define I2C_F_ICR(x)             (((uint8_t)((uint8_t)(x)<<I2C_F_ICR_SHIFT))&I2C_F_ICR_MASK)

#define I2C_F_MULT_MASK          0xC0u
#define I2C_F_MULT_SHIFT         6
#define I2C_F_MULT_WIDTH         2
#define I2C_F_MULT(x)             (((uint8_t)((uint8_t)(x)<<I2C_F_MULT_SHIFT))&I2C_F_MULT_MASK)

/* C1 Bit Fields */

#define I2C_C1_DMAEN_MASK        0x1u
#define I2C_C1_DMAEN_SHIFT       0
#define I2C_C1_DMAEN_WIDTH       1
#define I2C_C1_DMAEN(x)           (((uint8_t)((uint8_t)(x)<<I2C_C1_DMAEN_SHIFT))&I2C_C1_DMAEN_MASK)

#define I2C_C1_WUEN_MASK          0x2u
#define I2C_C1_WUEN_SHIFT         1
#define I2C_C1_WUEN_WIDTH         1
#define I2C_C1_WUEN(x)             (((uint8_t)((uint8_t)(x)<<I2C_C1_WUEN_SHIFT))&I2C_C1_WUEN_MASK)

#define I2C_C1_RSTA_MASK          0x4u
#define I2C_C1_RSTA_SHIFT         2
#define I2C_C1_RSTA_WIDTH         1

```

```

#define I2C_C1_RSTA(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_RSTA_SHIFT))&I2C_C1_RSTA_MASK)

#define I2C_C1_TXAK_MASK          0x8u
#define I2C_C1_TXAK_SHIFT         3
#define I2C_C1_TXAK_WIDTH         1

#define I2C_C1_TXAK(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_TXAK_SHIFT))&I2C_C1_TXAK_MASK)

#define I2C_C1_TX_MASK           0x10u
#define I2C_C1_TX_SHIFT          4
#define I2C_C1_TX_WIDTH          1

#define I2C_C1_TX(x)             (((uint8_t)((uint8_t)(x)<<I2C_C1_TX_SHIFT))&I2C_C1_TX_MASK)

#define I2C_C1_MST_MASK          0x20u
#define I2C_C1_MST_SHIFT         5
#define I2C_C1_MST_WIDTH         1

#define I2C_C1_MST(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_MST_SHIFT))&I2C_C1_MST_MASK)

#define I2C_C1_IICIE_MASK         0x40u
#define I2C_C1_IICIE_SHIFT        6
#define I2C_C1_IICIE_WIDTH        1

#define I2C_C1_IICIE(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_IICIE_SHIFT))&I2C_C1_IICIE_MASK)

#define I2C_C1_IICEN_MASK         0x80u
#define I2C_C1_IICEN_SHIFT        7
#define I2C_C1_IICEN_WIDTH        1

#define I2C_C1_IICEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_IICEN_SHIFT))&I2C_C1_IICEN_MASK)

/* S Bit Fields */

#define I2C_S_RXAK_MASK           0x1u
#define I2C_S_RXAK_SHIFT          0
#define I2C_S_RXAK_WIDTH          1

```

```

#define I2C_S_RXAK(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_RXAK_SHIFT))&I2C_S_RXAK_MASK)

#define I2C_S_IICIF_MASK          0x2u
#define I2C_S_IICIF_SHIFT         1
#define I2C_S_IICIF_WIDTH         1
#define I2C_S_IICIF(x)            (((uint8_t)((uint8_t)(x)<<I2C_S_IICIF_SHIFT))&I2C_S_IICIF_MASK)

#define I2C_S_SRW_MASK            0x4u
#define I2C_S_SRW_SHIFT           2
#define I2C_S_SRW_WIDTH           1
#define I2C_S_SRW(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_SRW_SHIFT))&I2C_S_SRW_MASK)

#define I2C_S_RAM_MASK             0x8u
#define I2C_S_RAM_SHIFT            3
#define I2C_S_RAM_WIDTH            1
#define I2C_S_RAM(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_RAM_SHIFT))&I2C_S_RAM_MASK)

#define I2C_S_ARBL_MASK            0x10u
#define I2C_S_ARBL_SHIFT           4
#define I2C_S_ARBL_WIDTH           1
#define I2C_S_ARBL(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_ARBL_SHIFT))&I2C_S_ARBL_MASK)

#define I2C_S_BUSY_MASK             0x20u
#define I2C_S_BUSY_SHIFT            5
#define I2C_S_BUSY_WIDTH            1
#define I2C_S_BUSY(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_BUSY_SHIFT))&I2C_S_BUSY_MASK)

#define I2C_S_IAAS_MASK             0x40u
#define I2C_S_IAAS_SHIFT            6
#define I2C_S_IAAS_WIDTH            1
#define I2C_S_IAAS(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_IAAS_SHIFT))&I2C_S_IAAS_MASK)

```

```

#define I2C_S_TCF_MASK           0x80u
#define I2C_S_TCF_SHIFT          7
#define I2C_S_TCF_WIDTH          1
#define I2C_S_TCF(x)             (((uint8_t)((uint8_t)(x))<<I2C_S_TCF_SHIFT))&I2C_S_TCF_MASK)

/* D Bit Fields */

#define I2C_D_DATA_MASK          0xFFu
#define I2C_D_DATA_SHIFT          0
#define I2C_D_DATA_WIDTH          8
#define I2C_D_DATA(x)             (((uint8_t)((uint8_t)(x))<<I2C_D_DATA_SHIFT))&I2C_D_DATA_MASK)

/* C2 Bit Fields */

#define I2C_C2_AD_MASK            0x7u
#define I2C_C2_AD_SHIFT           0
#define I2C_C2_AD_WIDTH           3
#define I2C_C2_AD(x)              (((uint8_t)((uint8_t)(x))<<I2C_C2_AD_SHIFT))&I2C_C2_AD_MASK)

#define I2C_C2_RMEN_MASK          0x8u
#define I2C_C2_RMEN_SHIFT          3
#define I2C_C2_RMEN_WIDTH          1
#define I2C_C2_RMEN(x)             (((uint8_t)((uint8_t)(x))<<I2C_C2_RMEN_SHIFT))&I2C_C2_RMEN_MASK)

#define I2C_C2_SBRC_MASK          0x10u
#define I2C_C2_SBRC_SHIFT          4
#define I2C_C2_SBRC_WIDTH          1
#define I2C_C2_SBRC(x)             (((uint8_t)((uint8_t)(x))<<I2C_C2_SBRC_SHIFT))&I2C_C2_SBRC_MASK)

#define I2C_C2_HDRS_MASK          0x20u
#define I2C_C2_HDRS_SHIFT          5
#define I2C_C2_HDRS_WIDTH          1
#define I2C_C2_HDRS(x)             (((uint8_t)((uint8_t)(x))<<I2C_C2_HDRS_SHIFT))&I2C_C2_HDRS_MASK)

```

```

#define I2C_C2_ADEXT_MASK           0x40u
#define I2C_C2_ADEXT_SHIFT         6
#define I2C_C2_ADEXT_WIDTH          1
#define I2C_C2_ADEXT(x) (((uint8_t)((uint8_t)(x)<<I2C_C2_ADEXT_SHIFT))&I2C_C2_ADEXT_MASK)
#define I2C_C2_GCAEN_MASK           0x80u
#define I2C_C2_GCAEN_SHIFT          7
#define I2C_C2_GCAEN_WIDTH          1
#define I2C_C2_GCAEN(x) (((uint8_t)((uint8_t)(x)<<I2C_C2_GCAEN_SHIFT))&I2C_C2_GCAEN_MASK)
/* FLT Bit Fields */
#define I2C_FLTFLT_MASK             0x1Fu
#define I2C_FLTFLT_SHIFT            0
#define I2C_FLTFLT_WIDTH             5
#define I2C_FLTFLT(x) (((uint8_t)((uint8_t)(x)<<I2C_FLTFLT_SHIFT))&I2C_FLTFLT_MASK)
#define I2C_FLT_STOPIE_MASK          0x20u
#define I2C_FLT_STOPIE_SHIFT          5
#define I2C_FLT_STOPIE_WIDTH          1
#define I2C_FLT_STOPIE(x) (((uint8_t)((uint8_t)(x)<<I2C_FLT_STOPIE_SHIFT))&I2C_FLT_STOPIE_MASK)
#define I2C_FLT_STOPF_MASK           0x40u
#define I2C_FLT_STOPF_SHIFT          6
#define I2C_FLT_STOPF_WIDTH           1
#define I2C_FLT_STOPF(x) (((uint8_t)((uint8_t)(x)<<I2C_FLT_STOPF_SHIFT))&I2C_FLT_STOPF_MASK)
#define I2C_FLT_SHEN_MASK            0x80u
#define I2C_FLT_SHEN_SHIFT           7
#define I2C_FLT_SHEN_WIDTH            1
#define I2C_FLT_SHEN(x) (((uint8_t)((uint8_t)(x)<<I2C_FLT_SHEN_SHIFT))&I2C_FLT_SHEN_MASK)

```

```

/* RA Bit Fields */

#define I2C_RA_RAD_MASK          0xFFu
#define I2C_RA_RAD_SHIFT         1
#define I2C_RA_RAD_WIDTH          7
#define I2C_RA_RAD(x) (((uint8_t)((uint8_t)(x)<<I2C_RA_RAD_SHIFT))&I2C_RA_RAD_MASK)

/* SMB Bit Fields */

#define I2C_SMB_SHTF2IE_MASK      0x1u
#define I2C_SMB_SHTF2IE_SHIFT       0
#define I2C_SMB_SHTF2IE_WIDTH       1
#define I2C_SMB_SHTF2IE(x) (((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2IE_SHIFT))&I2C_SMB_SHTF2IE_MASK)

#define I2C_SMB_SHTF2_MASK         0x2u
#define I2C_SMB_SHTF2_SHIFT         1
#define I2C_SMB_SHTF2_WIDTH          1
#define I2C_SMB_SHTF2(x) (((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2_SHIFT))&I2C_SMB_SHTF2_MASK)

#define I2C_SMB_SHTF1_MASK         0x4u
#define I2C_SMB_SHTF1_SHIFT         2
#define I2C_SMB_SHTF1_WIDTH          1
#define I2C_SMB_SHTF1(x) (((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF1_SHIFT))&I2C_SMB_SHTF1_MASK)

#define I2C_SMB_SLTF_MASK          0x8u
#define I2C_SMB_SLTF_SHIFT         3
#define I2C_SMB_SLTF_WIDTH          1
#define I2C_SMB_SLTF(x) (((uint8_t)((uint8_t)(x)<<I2C_SMB_SLTF_SHIFT))&I2C_SMB_SLTF_MASK)

#define I2C_SMB_TCKSEL_MASK        0x10u
#define I2C_SMB_TCKSEL_SHIFT        4
#define I2C_SMB_TCKSEL_WIDTH         1

```

```

#define I2C_SMB_TCKSEL(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_TCKSEL_SHIFT))&I2C_SMB_TCKSEL_MASK

#define I2C_SMB_SIICAEN_MASK          0x20u
#define I2C_SMB_SIICAEN_SHIFT        5
#define I2C_SMB_SIICAEN_WIDTH        1

#define I2C_SMB_SIICAEN(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_SIICAEN_SHIFT))&I2C_SMB_SIICAEN_MASK

#define I2C_SMB_ALERTEN_MASK         0x40u
#define I2C_SMB_ALERTEN_SHIFT        6
#define I2C_SMB_ALERTEN_WIDTH        1

#define I2C_SMB_ALERTEN(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_ALERTEN_SHIFT))&I2C_SMB_ALERTEN_MASK

#define I2C_SMB_FACK_MASK           0x80u
#define I2C_SMB_FACK_SHIFT          7
#define I2C_SMB_FACK_WIDTH          1

#define I2C_SMB_FACK(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_FACK_SHIFT))&I2C_SMB_FACK_MASK)

/* A2 Bit Fields */

#define I2C_A2_SAD_MASK             0xFEu
#define I2C_A2_SAD_SHIFT            1
#define I2C_A2_SAD_WIDTH            7

#define I2C_A2_SAD(x)
(((uint8_t)((uint8_t)(x))<<I2C_A2_SAD_SHIFT))&I2C_A2_SAD_MASK

/* SLTH Bit Fields */

#define I2C_SLTH_SSLT_MASK          0xFFu
#define I2C_SLTH_SSLT_SHIFT         0
#define I2C_SLTH_SSLT_WIDTH         8

#define I2C_SLTH_SSLT(x)
(((uint8_t)((uint8_t)(x))<<I2C_SLTH_SSLT_SHIFT))&I2C_SLTH_SSLT_MASK

/* SLTL Bit Fields */

#define I2C_SLTL_SSLT_MASK          0xFFu

```

```

#define I2C_SLTL_SSLT_SHIFT          0
#define I2C_SLTL_SSLT_WIDTH          8
#define I2C_SLTL_SSLT(x)
(((uint8_t)((uint8_t)(x)<<I2C_SLTL_SSLT_SHIFT))&I2C_SLTL_SSLT_MASK)

/*
 * @}
 */

/* /* end of group I2C_Register_Masks */

/* I2C - Peripheral instance base addresses */
/** Peripheral I2C0 base address */
#define I2C0_BASE                  (0x40066000u)
/** Peripheral I2C0 base pointer */
#define I2C0                      ((I2C_Type *)I2C0_BASE)
#define I2C0_BASE_PTR              (I2C0)

/** Peripheral I2C1 base address */
#define I2C1_BASE                  (0x40067000u)
/** Peripheral I2C1 base pointer */
#define I2C1                      ((I2C_Type *)I2C1_BASE)
#define I2C1_BASE_PTR              (I2C1)

/** Array initializer of I2C peripheral base addresses */
#define I2C_BASE_ADDRS             { I2C0_BASE, I2C1_BASE }

/** Array initializer of I2C peripheral base pointers */
#define I2C_BASE_PTRS              { I2C0, I2C1 }

/*
-- I2C - Register accessor macros
----- */

```

```

/*!

* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros

* @{

*/



/* I2C - Register instance definitions */

/* I2C0 */

#define I2C0_A1           I2C_A1_REG(I2C0)
#define I2C0_F            I2C_F_REG(I2C0)
#define I2C0_C1           I2C_C1_REG(I2C0)
#define I2C0_S            I2C_S_REG(I2C0)
#define I2C0_D            I2C_D_REG(I2C0)
#define I2C0_C2           I2C_C2_REG(I2C0)
#define I2C0_FLT          I2C_FLT_REG(I2C0)
#define I2C0_RA           I2C_RA_REG(I2C0)
#define I2C0_SMB          I2C_SMB_REG(I2C0)
#define I2C0_A2           I2C_A2_REG(I2C0)
#define I2C0_SLTH         I2C_SLTH_REG(I2C0)
#define I2C0_SLTL         I2C_SLTL_REG(I2C0)

/* I2C1 */

#define I2C1_A1           I2C_A1_REG(I2C1)
#define I2C1_F            I2C_F_REG(I2C1)
#define I2C1_C1           I2C_C1_REG(I2C1)
#define I2C1_S            I2C_S_REG(I2C1)
#define I2C1_D            I2C_D_REG(I2C1)
#define I2C1_C2           I2C_C2_REG(I2C1)
#define I2C1_FLT          I2C_FLT_REG(I2C1)

```

```
#define I2C1_RA           I2C_RA_REG(I2C1)
#define I2C1_SMB          I2C_SMB_REG(I2C1)
#define I2C1_A2            I2C_A2_REG(I2C1)
#define I2C1_SLTH          I2C_SLTH_REG(I2C1)
#define I2C1_SLTL          I2C_SLTL_REG(I2C1)
```

```
/*!
 * @}
 */
/* end of group I2C_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* end of group I2C_Peripheral_Access_Layer */
```

```
/*
-----  
-- LLWU Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup LLWU_Peripheral_Access_Layer LLWU Peripheral Access Layer  
 * @{
 */

```

```
/** LLWU - Register Layout Typedef */  
  
typedef struct {  
    __IO uint8_t PE1;           /**< LLWU Pin Enable 1 register, offset: 0x0 */  
    __IO uint8_t PE2;           /**< LLWU Pin Enable 2 register, offset: 0x1 */  
};
```

```

__IO uint8_t PE3;           /**< LLWU Pin Enable 3 register, offset: 0x2 */
__IO uint8_t PE4;           /**< LLWU Pin Enable 4 register, offset: 0x3 */
__IO uint8_t ME;            /**< LLWU Module Enable register, offset: 0x4 */
__IO uint8_t F1;            /**< LLWU Flag 1 register, offset: 0x5 */
__IO uint8_t F2;            /**< LLWU Flag 2 register, offset: 0x6 */
__I uint8_t F3;             /**< LLWU Flag 3 register, offset: 0x7 */
__IO uint8_t FILT1;         /**< LLWU Pin Filter 1 register, offset: 0x8 */
__IO uint8_t FILT2;         /**< LLWU Pin Filter 2 register, offset: 0x9 */

} LLWU_Type, *LLWU_MemMapPtr;

```

```

/*
-----*
-- LLWU - Register accessor macros
----- */

```

```

/*!
* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros
* @{
*/

```

```

/* LLWU - Register accessors */

#define LLWU_PE1_REG(base)      ((base)->PE1)
#define LLWU_PE2_REG(base)      ((base)->PE2)
#define LLWU_PE3_REG(base)      ((base)->PE3)
#define LLWU_PE4_REG(base)      ((base)->PE4)
#define LLWU_ME_REG(base)       ((base)->ME)
#define LLWU_F1_REG(base)       ((base)->F1)
#define LLWU_F2_REG(base)       ((base)->F2)
#define LLWU_F3_REG(base)       ((base)->F3)

```

```
#define LLWU_FILT1_REG(base) ((base)->FILT1)
#define LLWU_FILT2_REG(base) ((base)->FILT2)
```

```
/*!
 * @}
 */
/* /* end of group LLWU_Register_Accessor_Macros */
```

```
/*
-----  
-- LLWU Register Masks  
----- */
```

```
/*!  
 * @addtogroup LLWU_Register_Masks LLWU Register Masks  
 * @{  
 */
```

```
/* PE1 Bit Fields */

#define LLWU_PE1_WUPE0_MASK      0x3u
#define LLWU_PE1_WUPE0_SHIFT     0
#define LLWU_PE1_WUPE0_WIDTH     2
#define LLWU_PE1_WUPE0(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE0_SHIFT))&LLWU_PE1_WUPE0_MASK)

#define LLWU_PE1_WUPE1_MASK      0xCu
#define LLWU_PE1_WUPE1_SHIFT     2
#define LLWU_PE1_WUPE1_WIDTH     2
#define LLWU_PE1_WUPE1(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE1_SHIFT))&LLWU_PE1_WUPE1_MASK)

#define LLWU_PE1_WUPE2_MASK      0x30u
```

```

#define LLWU_PE1_WUPE2_SHIFT          4
#define LLWU_PE1_WUPE2_WIDTH          2
#define LLWU_PE1_WUPE2(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE2_SHIFT))&LLWU_PE1_WUPE2_MASK)
#define LLWU_PE1_WUPE3_MASK           0xC0u
#define LLWU_PE1_WUPE3_SHIFT          6
#define LLWU_PE1_WUPE3_WIDTH          2
#define LLWU_PE1_WUPE3(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE3_SHIFT))&LLWU_PE1_WUPE3_MASK)

/* PE2 Bit Fields */

#define LLWU_PE2_WUPE4_MASK           0x3u
#define LLWU_PE2_WUPE4_SHIFT          0
#define LLWU_PE2_WUPE4_WIDTH          2
#define LLWU_PE2_WUPE4(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE4_SHIFT))&LLWU_PE2_WUPE4_MASK)
#define LLWU_PE2_WUPE5_MASK           0xCu
#define LLWU_PE2_WUPE5_SHIFT          2
#define LLWU_PE2_WUPE5_WIDTH          2
#define LLWU_PE2_WUPE5(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE5_SHIFT))&LLWU_PE2_WUPE5_MASK)
#define LLWU_PE2_WUPE6_MASK           0x30u
#define LLWU_PE2_WUPE6_SHIFT          4
#define LLWU_PE2_WUPE6_WIDTH          2
#define LLWU_PE2_WUPE6(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE6_SHIFT))&LLWU_PE2_WUPE6_MASK)
#define LLWU_PE2_WUPE7_MASK           0xC0u
#define LLWU_PE2_WUPE7_SHIFT          6
#define LLWU_PE2_WUPE7_WIDTH          2
#define LLWU_PE2_WUPE7(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE7_SHIFT))&LLWU_PE2_WUPE7_MASK)

/* PE3 Bit Fields */

```

```

#define LLWU_PE3_WUPE8_MASK          0x3u
#define LLWU_PE3_WUPE8_SHIFT         0
#define LLWU_PE3_WUPE8_WIDTH         2
#define LLWU_PE3_WUPE8(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE8_SHIFT))&LLWU_PE3_WUPE8_MASK)

#define LLWU_PE3_WUPE9_MASK          0xCu
#define LLWU_PE3_WUPE9_SHIFT         2
#define LLWU_PE3_WUPE9_WIDTH         2
#define LLWU_PE3_WUPE9(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE9_SHIFT))&LLWU_PE3_WUPE9_MASK)

#define LLWU_PE3_WUPE10_MASK         0x30u
#define LLWU_PE3_WUPE10_SHIFT        4
#define LLWU_PE3_WUPE10_WIDTH        2
#define LLWU_PE3_WUPE10(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE10_SHIFT))&LLWU_PE3_WUPE10_MASK)

#define LLWU_PE3_WUPE11_MASK         0xC0u
#define LLWU_PE3_WUPE11_SHIFT        6
#define LLWU_PE3_WUPE11_WIDTH        2
#define LLWU_PE3_WUPE11(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE11_SHIFT))&LLWU_PE3_WUPE11_MASK)

/* PE4 Bit Fields */

#define LLWU_PE4_WUPE12_MASK         0x3u
#define LLWU_PE4_WUPE12_SHIFT        0
#define LLWU_PE4_WUPE12_WIDTH        2
#define LLWU_PE4_WUPE12(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE12_SHIFT))&LLWU_PE4_WUPE12_MASK)

#define LLWU_PE4_WUPE13_MASK         0xCu
#define LLWU_PE4_WUPE13_SHIFT        2
#define LLWU_PE4_WUPE13_WIDTH        2
#define LLWU_PE4_WUPE13(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE13_SHIFT))&LLWU_PE4_WUPE13_MASK)

```

```

#define LLWU_PE4_WUPE14_MASK          0x30u
#define LLWU_PE4_WUPE14_SHIFT         4
#define LLWU_PE4_WUPE14_WIDTH         2
#define LLWU_PE4_WUPE14(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE14_SHIFT))&LLWU_PE4_WUPE14_MASK

#define LLWU_PE4_WUPE15_MASK          0xC0u
#define LLWU_PE4_WUPE15_SHIFT         6
#define LLWU_PE4_WUPE15_WIDTH         2
#define LLWU_PE4_WUPE15(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE15_SHIFT))&LLWU_PE4_WUPE15_MASK

/* ME Bit Fields */

#define LLWU_ME_WUME0_MASK           0x1u
#define LLWU_ME_WUME0_SHIFT          0
#define LLWU_ME_WUME0_WIDTH          1
#define LLWU_ME_WUME0(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME0_SHIFT))&LLWU_ME_WUME0_MASK

#define LLWU_ME_WUME1_MASK           0x2u
#define LLWU_ME_WUME1_SHIFT          1
#define LLWU_ME_WUME1_WIDTH          1
#define LLWU_ME_WUME1(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME1_SHIFT))&LLWU_ME_WUME1_MASK

#define LLWU_ME_WUME2_MASK           0x4u
#define LLWU_ME_WUME2_SHIFT          2
#define LLWU_ME_WUME2_WIDTH          1
#define LLWU_ME_WUME2(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME2_SHIFT))&LLWU_ME_WUME2_MASK

#define LLWU_ME_WUME3_MASK           0x8u
#define LLWU_ME_WUME3_SHIFT          3
#define LLWU_ME_WUME3_WIDTH          1
#define LLWU_ME_WUME3(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME3_SHIFT))&LLWU_ME_WUME3_MASK

```

```

#define LLWU_ME_WUME4_MASK          0x10u
#define LLWU_ME_WUME4_SHIFT         4
#define LLWU_ME_WUME4_WIDTH          1
#define LLWU_ME_WUME4(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME4_SHIFT))&LLWU_ME_WUME4_MASK)

#define LLWU_ME_WUME5_MASK          0x20u
#define LLWU_ME_WUME5_SHIFT         5
#define LLWU_ME_WUME5_WIDTH          1
#define LLWU_ME_WUME5(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME5_SHIFT))&LLWU_ME_WUME5_MASK)

#define LLWU_ME_WUME6_MASK          0x40u
#define LLWU_ME_WUME6_SHIFT         6
#define LLWU_ME_WUME6_WIDTH          1
#define LLWU_ME_WUME6(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME6_SHIFT))&LLWU_ME_WUME6_MASK)

#define LLWU_ME_WUME7_MASK          0x80u
#define LLWU_ME_WUME7_SHIFT         7
#define LLWU_ME_WUME7_WIDTH          1
#define LLWU_ME_WUME7(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME7_SHIFT))&LLWU_ME_WUME7_MASK)

/* F1 Bit Fields */

#define LLWU_F1_WUFO_MASK           0x1u
#define LLWU_F1_WUFO_SHIFT          0
#define LLWU_F1_WUFO_WIDTH          1
#define LLWU_F1_WUFO(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUFO_SHIFT))&LLWU_F1_WUFO_MASK)

#define LLWU_F1_WUF1_MASK           0x2u
#define LLWU_F1_WUF1_SHIFT          1
#define LLWU_F1_WUF1_WIDTH          1
#define LLWU_F1_WUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF1_SHIFT))&LLWU_F1_WUF1_MASK)

```

```

#define LLWU_F1_WUF2_MASK          0x4u
#define LLWU_F1_WUF2_SHIFT         2
#define LLWU_F1_WUF2_WIDTH         1
#define LLWU_F1_WUF2(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF2_SHIFT))&LLWU_F1_WUF2_MASK)

#define LLWU_F1_WUF3_MASK          0x8u
#define LLWU_F1_WUF3_SHIFT         3
#define LLWU_F1_WUF3_WIDTH         1
#define LLWU_F1_WUF3(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF3_SHIFT))&LLWU_F1_WUF3_MASK)

#define LLWU_F1_WUF4_MASK          0x10u
#define LLWU_F1_WUF4_SHIFT         4
#define LLWU_F1_WUF4_WIDTH         1
#define LLWU_F1_WUF4(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF4_SHIFT))&LLWU_F1_WUF4_MASK)

#define LLWU_F1_WUF5_MASK          0x20u
#define LLWU_F1_WUF5_SHIFT         5
#define LLWU_F1_WUF5_WIDTH         1
#define LLWU_F1_WUF5(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF5_SHIFT))&LLWU_F1_WUF5_MASK)

#define LLWU_F1_WUF6_MASK          0x40u
#define LLWU_F1_WUF6_SHIFT         6
#define LLWU_F1_WUF6_WIDTH         1
#define LLWU_F1_WUF6(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF6_SHIFT))&LLWU_F1_WUF6_MASK)

#define LLWU_F1_WUF7_MASK          0x80u
#define LLWU_F1_WUF7_SHIFT         7
#define LLWU_F1_WUF7_WIDTH         1
#define LLWU_F1_WUF7(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF7_SHIFT))&LLWU_F1_WUF7_MASK)

/* F2 Bit Fields */

```

```

#define LLWU_F2_WUF8_MASK          0x1u
#define LLWU_F2_WUF8_SHIFT         0
#define LLWU_F2_WUF8_WIDTH         1
#define LLWU_F2_WUF8(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF8_SHIFT))&LLWU_F2_WUF8_MASK

#define LLWU_F2_WUF9_MASK          0x2u
#define LLWU_F2_WUF9_SHIFT         1
#define LLWU_F2_WUF9_WIDTH         1
#define LLWU_F2_WUF9(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF9_SHIFT))&LLWU_F2_WUF9_MASK

#define LLWU_F2_WUF10_MASK         0x4u
#define LLWU_F2_WUF10_SHIFT        2
#define LLWU_F2_WUF10_WIDTH        1
#define LLWU_F2_WUF10(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF10_SHIFT))&LLWU_F2_WUF10_MASK

#define LLWU_F2_WUF11_MASK         0x8u
#define LLWU_F2_WUF11_SHIFT        3
#define LLWU_F2_WUF11_WIDTH        1
#define LLWU_F2_WUF11(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF11_SHIFT))&LLWU_F2_WUF11_MASK

#define LLWU_F2_WUF12_MASK         0x10u
#define LLWU_F2_WUF12_SHIFT        4
#define LLWU_F2_WUF12_WIDTH        1
#define LLWU_F2_WUF12(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF12_SHIFT))&LLWU_F2_WUF12_MASK

#define LLWU_F2_WUF13_MASK         0x20u
#define LLWU_F2_WUF13_SHIFT        5
#define LLWU_F2_WUF13_WIDTH        1
#define LLWU_F2_WUF13(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF13_SHIFT))&LLWU_F2_WUF13_MASK

#define LLWU_F2_WUF14_MASK         0x40u

```

```

#define LLWU_F2_WUF14_SHIFT          6
#define LLWU_F2_WUF14_WIDTH          1
#define LLWU_F2_WUF14(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF14_SHIFT))&LLWU_F2_WUF14_MASK)

#define LLWU_F2_WUF15_MASK          0x80u
#define LLWU_F2_WUF15_SHIFT          7
#define LLWU_F2_WUF15_WIDTH          1
#define LLWU_F2_WUF15(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF15_SHIFT))&LLWU_F2_WUF15_MASK)

/* F3 Bit Fields */

#define LLWU_F3_MWUF0_MASK          0x1u
#define LLWU_F3_MWUF0_SHIFT          0
#define LLWU_F3_MWUF0_WIDTH          1
#define LLWU_F3_MWUF0(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF0_SHIFT))&LLWU_F3_MWUF0_MASK)

#define LLWU_F3_MWUF1_MASK          0x2u
#define LLWU_F3_MWUF1_SHIFT          1
#define LLWU_F3_MWUF1_WIDTH          1
#define LLWU_F3_MWUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF1_SHIFT))&LLWU_F3_MWUF1_MASK)

#define LLWU_F3_MWUF2_MASK          0x4u
#define LLWU_F3_MWUF2_SHIFT          2
#define LLWU_F3_MWUF2_WIDTH          1
#define LLWU_F3_MWUF2(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF2_SHIFT))&LLWU_F3_MWUF2_MASK)

#define LLWU_F3_MWUF3_MASK          0x8u
#define LLWU_F3_MWUF3_SHIFT          3
#define LLWU_F3_MWUF3_WIDTH          1
#define LLWU_F3_MWUF3(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF3_SHIFT))&LLWU_F3_MWUF3_MASK)

#define LLWU_F3_MWUF4_MASK          0x10u

```

```

#define LLWU_F3_MWUF4_SHIFT          4
#define LLWU_F3_MWUF4_WIDTH          1
#define LLWU_F3_MWUF4(x) (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF4_SHIFT))&LLWU_F3_MWUF4_MASK)
#define LLWU_F3_MWUF5_MASK           0x20u
#define LLWU_F3_MWUF5_SHIFT          5
#define LLWU_F3_MWUF5_WIDTH          1
#define LLWU_F3_MWUF5(x) (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF5_SHIFT))&LLWU_F3_MWUF5_MASK)
#define LLWU_F3_MWUF6_MASK           0x40u
#define LLWU_F3_MWUF6_SHIFT          6
#define LLWU_F3_MWUF6_WIDTH          1
#define LLWU_F3_MWUF6(x) (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF6_SHIFT))&LLWU_F3_MWUF6_MASK)
#define LLWU_F3_MWUF7_MASK           0x80u
#define LLWU_F3_MWUF7_SHIFT          7
#define LLWU_F3_MWUF7_WIDTH          1
#define LLWU_F3_MWUF7(x) (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF7_SHIFT))&LLWU_F3_MWUF7_MASK)

/* FILT1 Bit Fields */

#define LLWU_FILT1_FILTSEL_MASK      0xFu
#define LLWU_FILT1_FILTSEL_SHIFT     0
#define LLWU_FILT1_FILTSEL_WIDTH     4
#define LLWU_FILT1_FILTSEL(x) (((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTSEL_SHIFT))&LLWU_FILT1_FILTSEL_MASK)
#define LLWU_FILT1_FILTE_MASK        0x60u
#define LLWU_FILT1_FILTE_SHIFT       5
#define LLWU_FILT1_FILTE_WIDTH       2
#define LLWU_FILT1_FILTE(x) (((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTE_SHIFT))&LLWU_FILT1_FILTE_MASK)
#define LLWU_FILT1_FILTF_MASK        0x80u

```

```

#define LLWU_FILT1_FILTF_SHIFT      7
#define LLWU_FILT1_FILTF_WIDTH      1
#define LLWU_FILT1_FILTF(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTF_SHIFT))&LLWU_FILT1_FILTF_MASK)

/* FILT2 Bit Fields */

#define LLWU_FILT2_FILTSEL_MASK      0xFu
#define LLWU_FILT2_FILTSEL_SHIFT      0
#define LLWU_FILT2_FILTSEL_WIDTH      4
#define LLWU_FILT2_FILTSEL(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTSEL_SHIFT))&LLWU_FILT2_FILTSEL_MASK)

#define LLWU_FILT2_FILTE_MASK        0x60u
#define LLWU_FILT2_FILTE_SHIFT        5
#define LLWU_FILT2_FILTE_WIDTH        2
#define LLWU_FILT2_FILTE(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTE_SHIFT))&LLWU_FILT2_FILTE_MASK)

#define LLWU_FILT2_FILTF_MASK        0x80u
#define LLWU_FILT2_FILTF_SHIFT        7
#define LLWU_FILT2_FILTF_WIDTH        1
#define LLWU_FILT2_FILTF(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTF_SHIFT))&LLWU_FILT2_FILTF_MASK)

*/
* @}
/* /* end of group LLWU_Register_Masks */

```

```

/* LLWU - Peripheral instance base addresses */
/** Peripheral LLWU base address */
#define LLWU_BASE                  (0x4007C000u)
/** Peripheral LLWU base pointer */

```

```

#define LLWU ((LLWU_Type *)LLWU_BASE)

#define LLWU_BASE_PTR (LLWU)

/** Array initializer of LLWU peripheral base addresses */

#define LLWU_BASE_ADDRS { LLWU }

/** Array initializer of LLWU peripheral base pointers */

#define LLWU_BASE_PTRS { LLWU }

/*
-- LLWU - Register accessor macros
*/
/*!
* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros
* @{
*/
/* LLWU - Register instance definitions */
/* LLWU */

#define LLWU_PE1 LLWU_PE1_REG(LLWU)
#define LLWU_PE2 LLWU_PE2_REG(LLWU)
#define LLWU_PE3 LLWU_PE3_REG(LLWU)
#define LLWU_PE4 LLWU_PE4_REG(LLWU)
#define LLWU_ME LLWU_ME_REG(LLWU)
#define LLWU_F1 LLWU_F1_REG(LLWU)
#define LLWU_F2 LLWU_F2_REG(LLWU)
#define LLWU_F3 LLWU_F3_REG(LLWU)
#define LLWU_FILT1 LLWU_FILT1_REG(LLWU)
#define LLWU_FILT2 LLWU_FILT2_REG(LLWU)

```

```
/*!  
 * @}  
 */ /* end of group LLWU_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group LLWU_Peripheral_Access_Layer */
```

```
/* -----  
-- LPTMR Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup LPTMR_Peripheral_Access_Layer LPTMR Peripheral Access Layer  
 * @{  
 */
```

```
/** LPTMR - Register Layout Typedef */  
  
typedef struct {  
    __IO uint32_t CSR;           /**< Low Power Timer Control Status Register, offset: 0x0 */  
    __IO uint32_t PSR;          /**< Low Power Timer Prescale Register, offset: 0x4 */  
    __IO uint32_t CMR;          /**< Low Power Timer Compare Register, offset: 0x8 */  
    __IO uint32_t CNR;          /**< Low Power Timer Counter Register, offset: 0xC */  
} LPTMR_Type, *LPTMR_MemMapPtr;
```

```
/* -----
```

```
-- LPTMR - Register accessor macros
----- */
/*!
* @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
* @{
*/
/* LPTMR - Register accessors */
#define LPTMR_CSR_REG(base) ((base)->CSR)
#define LPTMR_PSR_REG(base) ((base)->PSR)
#define LPTMR_CMR_REG(base) ((base)->CMR)
#define LPTMR_CNR_REG(base) ((base)->CNR)

/*!
* @}
*/
/* /* end of group LPTMR_Register_Accessor_Macros */

/*
----- --
-- LPTMR Register Masks
----- */
/*!
* @addtogroup LPTMR_Register_Masks LPTMR Register Masks
* @{
*/

```

```

/* CSR Bit Fields */

#define LPTMR_CSR_TEN_MASK          0x1u
#define LPTMR_CSR_TEN_SHIFT         0
#define LPTMR_CSR_TEN_WIDTH         1
#define LPTMR_CSR_TEN(x) (((uint32_t)(x))<<LPTMR_CSR_TEN_SHIFT)&LPTMR_CSR_TEN_MASK)

#define LPTMR_CSR_TMS_MASK          0x2u
#define LPTMR_CSR_TMS_SHIFT         1
#define LPTMR_CSR_TMS_WIDTH         1
#define LPTMR_CSR_TMS(x) (((uint32_t)(x))<<LPTMR_CSR_TMS_SHIFT)&LPTMR_CSR_TMS_MASK)

#define LPTMR_CSR_TFC_MASK          0x4u
#define LPTMR_CSR_TFC_SHIFT         2
#define LPTMR_CSR_TFC_WIDTH         1
#define LPTMR_CSR_TFC(x) (((uint32_t)(x))<<LPTMR_CSR_TFC_SHIFT)&LPTMR_CSR_TFC_MASK)

#define LPTMR_CSR TPP_MASK          0x8u
#define LPTMR_CSR TPP_SHIFT         3
#define LPTMR_CSR TPP_WIDTH         1
#define LPTMR_CSR TPP(x) (((uint32_t)(x))<<LPTMR_CSR TPP_SHIFT)&LPTMR_CSR TPP_MASK)

#define LPTMR_CSR TPS_MASK          0x30u
#define LPTMR_CSR TPS_SHIFT         4
#define LPTMR_CSR TPS_WIDTH         2
#define LPTMR_CSR TPS(x) (((uint32_t)(x))<<LPTMR_CSR TPS_SHIFT)&LPTMR_CSR TPS_MASK)

#define LPTMR_CSR TIE_MASK          0x40u
#define LPTMR_CSR TIE_SHIFT         6
#define LPTMR_CSR TIE_WIDTH         1
#define LPTMR_CSR TIE(x) (((uint32_t)(x))<<LPTMR_CSR TIE_SHIFT)&LPTMR_CSR TIE_MASK)

```

```

#define LPTMR_CSR_TCF_MASK           0x80u
#define LPTMR_CSR_TCF_SHIFT          7
#define LPTMR_CSR_TCF_WIDTH          1
#define LPTMR_CSR_TCF(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR_TCF_SHIFT))&LPTMR_CSR_TCF_MASK)

/* PSR Bit Fields */

#define LPTMR_PSR_PCS_MASK           0x3u
#define LPTMR_PSR_PCS_SHIFT          0
#define LPTMR_PSR_PCS_WIDTH          2
#define LPTMR_PSR_PCS(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PCS_SHIFT))&LPTMR_PSR_PCS_MASK)

#define LPTMR_PSR_PBYP_MASK           0x4u
#define LPTMR_PSR_PBYP_SHIFT          2
#define LPTMR_PSR_PBYP_WIDTH          1
#define LPTMR_PSR_PBYP(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PBYP_SHIFT))&LPTMR_PSR_PBYP_MASK)

#define LPTMR_PSR_PRESCALE_MASK        0x78u
#define LPTMR_PSR_PRESCALE_SHIFT       3
#define LPTMR_PSR_PRESCALE_WIDTH       4
#define LPTMR_PSR_PRESCALE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PRESCALE_SHIFT))&LPTMR_PSR_PRESCALE_MASK)

/* CMR Bit Fields */

#define LPTMR_CMR_COMPARE_MASK         0xFFFFu
#define LPTMR_CMR_COMPARE_SHIFT        0
#define LPTMR_CMR_COMPARE_WIDTH        16
#define LPTMR_CMR_COMPARE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CMR_COMPARE_SHIFT))&LPTMR_CMR_COMPARE_MASK)

/* CNR Bit Fields */

#define LPTMR_CNR_COUNTER_MASK         0xFFFFu
#define LPTMR_CNR_COUNTER_SHIFT        0

```

```

#define LPTMR_CNR_COUNTER_WIDTH          16

#define LPTMR_CNR_COUNTER(x)
(((uint32_t) (((uint32_t)(x))<<LPTMR_CNR_COUNTER_SHIFT))&LPTMR_CNR_COUNTER_MASK)

/*
 * @}
 */

/* /* end of group LPTMR_Register_Masks */

/* LPTMR - Peripheral instance base addresses */

/** Peripheral LPTMRO base address */

#define LPTMRO_BASE          (0x40040000u)

/** Peripheral LPTMRO base pointer */

#define LPTMRO          ((LPTMR_Type *)LPTMRO_BASE)

#define LPTMRO_BASE_PTR      (LPTMRO)

/** Array initializer of LPTMR peripheral base addresses */

#define LPTMR_BASE_ADDRS     { LPTMRO_BASE }

/** Array initializer of LPTMR peripheral base pointers */

#define LPTMR_BASE_PTRS      { LPTMRO }

/*
-----*
-- LPTMR - Register accessor macros
----- */

```

```

/*
 * @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
 * @{
 */


```

```
/* LPTMR - Register instance definitions */

/* LPTMRO */

#define LPTMRO_CSR           LPTMR_CSR_REG(LPTMRO)
#define LPTMRO_PSR           LPTMR_PSR_REG(LPTMRO)
#define LPTMRO_CMR           LPTMR_CMR_REG(LPTMRO)
#define LPTMRO_CNR           LPTMR_CNR_REG(LPTMRO)

/*!
 * @}

 */ /* end of group LPTMR_Register_Accessor_Macros */
```

```
/*!
 * @}

 */ /* end of group LPTMR_Peripheral_Access_Layer */
```

```
/* -----
-- MCG Peripheral Access Layer
----- */
```

```
/*!
 * @addtogroup MCG_Peripheral_Access_Layer MCG Peripheral Access Layer
 * @{
 */


```

```
/** MCG - Register Layout Typedef */

typedef struct {
```

```

__IO uint8_t C1;           /**< MCG Control 1 Register, offset: 0x0 */
__IO uint8_t C2;           /**< MCG Control 2 Register, offset: 0x1 */
__IO uint8_t C3;           /**< MCG Control 3 Register, offset: 0x2 */
__IO uint8_t C4;           /**< MCG Control 4 Register, offset: 0x3 */
__IO uint8_t C5;           /**< MCG Control 5 Register, offset: 0x4 */
__IO uint8_t C6;           /**< MCG Control 6 Register, offset: 0x5 */
__IO uint8_t S;            /**< MCG Status Register, offset: 0x6 */

    uint8_t RESERVED_0[1];

__IO uint8_t SC;           /**< MCG Status and Control Register, offset: 0x8 */

    uint8_t RESERVED_1[1];

__IO uint8_t ATCVH;         /**< MCG Auto Trim Compare Value High Register, offset: 0xA */
__IO uint8_t ATCVL;         /**< MCG Auto Trim Compare Value Low Register, offset: 0xB */

__I uint8_t C7;            /**< MCG Control 7 Register, offset: 0xC */
__IO uint8_t C8;            /**< MCG Control 8 Register, offset: 0xD */
__I uint8_t C9;            /**< MCG Control 9 Register, offset: 0xE */
__I uint8_t C10;           /**< MCG Control 10 Register, offset: 0xF */

} MCG_Type, *MCG_MemMapPtr;

```

```

/*
-----*
-- MCG - Register accessor macros
----- */

```

```

/*!
* @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros
* @{
*/

```

```

/* MCG - Register accessors */

```

```
#define MCG_C1_REG(base)          ((base)->C1)
#define MCG_C2_REG(base)          ((base)->C2)
#define MCG_C3_REG(base)          ((base)->C3)
#define MCG_C4_REG(base)          ((base)->C4)
#define MCG_C5_REG(base)          ((base)->C5)
#define MCG_C6_REG(base)          ((base)->C6)
#define MCG_S_REG(base)           ((base)->S)
#define MCG_SC_REG(base)          ((base)->SC)
#define MCG_ATCVH_REG(base)        ((base)->ATCVH)
#define MCG_ATCVL_REG(base)        ((base)->ATCVL)
#define MCG_C7_REG(base)          ((base)->C7)
#define MCG_C8_REG(base)          ((base)->C8)
#define MCG_C9_REG(base)          ((base)->C9)
#define MCG_C10_REG(base)         ((base)->C10)
```

```
/*
 * @}
 */
/* /* end of group MCG_Register_Accessor_Macros */
```

```
/*
-----  
-- MCG Register Masks  
----- */
```

```
/*
 * @addtogroup MCG_Register_Masks MCG Register Masks
 * @{
 */
*/
```

```

/* C1 Bit Fields */

#define MCG_C1_IREFSTEN_MASK          0x1u
#define MCG_C1_IREFSTEN_SHIFT         0
#define MCG_C1_IREFSTEN_WIDTH         1
#define MCG_C1_IREFSTEN(x)            (((uint8_t)((uint8_t)(x)<<MCG_C1_IREFSTEN_SHIFT))&MCG_C1_IREFSTEN_MASK)

#define MCG_C1_IRCLKEN_MASK           0x2u
#define MCG_C1_IRCLKEN_SHIFT          1
#define MCG_C1_IRCLKEN_WIDTH          1
#define MCG_C1_IRCLKEN(x)             (((uint8_t)((uint8_t)(x)<<MCG_C1_IRCLKEN_SHIFT))&MCG_C1_IRCLKEN_MASK)

#define MCG_C1_IREFS_MASK              0x4u
#define MCG_C1_IREFS_SHIFT             2
#define MCG_C1_IREFS_WIDTH             1
#define MCG_C1_IREFS(x)                (((uint8_t)((uint8_t)(x)<<MCG_C1_IREFS_SHIFT))&MCG_C1_IREFS_MASK)

#define MCG_C1_FRDIV_MASK              0x38u
#define MCG_C1_FRDIV_SHIFT              3
#define MCG_C1_FRDIV_WIDTH              3
#define MCG_C1_FRDIV(x)                (((uint8_t)((uint8_t)(x)<<MCG_C1_FRDIV_SHIFT))&MCG_C1_FRDIV_MASK)

#define MCG_C1_CLKS_MASK                0xC0u
#define MCG_C1_CLKS_SHIFT                  6
#define MCG_C1_CLKS_WIDTH                  2
#define MCG_C1_CLKS(x)                    (((uint8_t)((uint8_t)(x)<<MCG_C1_CLKS_SHIFT))&MCG_C1_CLKS_MASK)

/* C2 Bit Fields */

#define MCG_C2_IRCS_MASK                0x1u
#define MCG_C2_IRCS_SHIFT                  0
#define MCG_C2_IRCS_WIDTH                  1

```

```

#define MCG_C2_IRCS(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_IRCS_SHIFT))&MCG_C2_IRCS_MASK)

#define MCG_C2_LP_MASK          0x2u
#define MCG_C2_LP_SHIFT         1
#define MCG_C2_LP_WIDTH         1

#define MCG_C2_LP(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_LP_SHIFT))&MCG_C2_LP_MASK)

#define MCG_C2_EREFS0_MASK       0x4u
#define MCG_C2_EREFS0_SHIFT      2
#define MCG_C2_EREFS0_WIDTH      1

#define MCG_C2_EREFS0(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_EREFS0_SHIFT))&MCG_C2_EREFS0_MASK)

#define MCG_C2_HGO0_MASK         0x8u
#define MCG_C2_HGO0_SHIFT        3
#define MCG_C2_HGO0_WIDTH        1

#define MCG_C2_HGO0(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_HGO0_SHIFT))&MCG_C2_HGO0_MASK)

#define MCG_C2_RANGE0_MASK        0x30u
#define MCG_C2_RANGE0_SHIFT       4
#define MCG_C2_RANGE0_WIDTH       2

#define MCG_C2_RANGE0(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_RANGE0_SHIFT))&MCG_C2_RANGE0_MASK)

#define MCG_C2_LOCRE0_MASK        0x80u
#define MCG_C2_LOCRE0_SHIFT       7
#define MCG_C2_LOCRE0_WIDTH       1

#define MCG_C2_LOCRE0(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C2_LOCRE0_SHIFT))&MCG_C2_LOCRE0_MASK)

/* C3 Bit Fields */

#define MCG_C3_SCTRIM_MASK        0xFFu
#define MCG_C3_SCTRIM_SHIFT       0
#define MCG_C3_SCTRIM_WIDTH       8

```

```

#define MCG_C3_SCTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C3_SCTRIM_SHIFT))&MCG_C3_SCTRIM_MASK)

/* C4 Bit Fields */

#define MCG_C4_SCFTRIM_MASK          0x1u
#define MCG_C4_SCFTRIM_SHIFT         0
#define MCG_C4_SCFTRIM_WIDTH         1

#define MCG_C4_SCFTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_SCFTRIM_SHIFT))&MCG_C4_SCFTRIM_MASK)

#define MCG_C4_FCTRIM_MASK          0x1Eu
#define MCG_C4_FCTRIM_SHIFT          1
#define MCG_C4_FCTRIM_WIDTH          4

#define MCG_C4_FCTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_FCTRIM_SHIFT))&MCG_C4_FCTRIM_MASK)

#define MCG_C4_DRST_DRS_MASK         0x60u
#define MCG_C4_DRST_DRS_SHIFT         5
#define MCG_C4_DRST_DRS_WIDTH         2

#define MCG_C4_DRST_DRS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_DRST_DRS_SHIFT))&MCG_C4_DRST_DRS_MASK)

#define MCG_C4_DMX32_MASK            0x80u
#define MCG_C4_DMX32_SHIFT             7
#define MCG_C4_DMX32_WIDTH              1

#define MCG_C4_DMX32(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_DMX32_SHIFT))&MCG_C4_DMX32_MASK)

/* C5 Bit Fields */

#define MCG_C5_PRDIV0_MASK           0x1Fu
#define MCG_C5_PRDIV0_SHIFT             0
#define MCG_C5_PRDIV0_WIDTH              5

#define MCG_C5_PRDIV0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C5_PRDIV0_SHIFT))&MCG_C5_PRDIV0_MASK)

#define MCG_C5_PLLSTENO_MASK          0x20u
#define MCG_C5_PLLSTENO_SHIFT          5

```

```

#define MCG_C5_PLLSTENO_WIDTH          1
#define MCG_C5_PLLSTENO(x) (((uint8_t)((uint8_t)(x)<<MCG_C5_PLLSTENO_SHIFT))&MCG_C5_PLLSTENO_MASK)
#define MCG_C5_PLLCLKENO_MASK          0x40u
#define MCG_C5_PLLCLKENO_SHIFT         6
#define MCG_C5_PLLCLKENO_WIDTH         1
#define MCG_C5_PLLCLKENO(x) (((uint8_t)((uint8_t)(x)<<MCG_C5_PLLCLKENO_SHIFT))&MCG_C5_PLLCLKENO_MASK)

/* C6 Bit Fields */

#define MCG_C6_VDIV0_MASK              0x1Fu
#define MCG_C6_VDIV0_SHIFT             0
#define MCG_C6_VDIV0_WIDTH             5
#define MCG_C6_VDIV0(x) (((uint8_t)((uint8_t)(x)<<MCG_C6_VDIV0_SHIFT))&MCG_C6_VDIV0_MASK)

#define MCG_C6_CME0_MASK               0x20u
#define MCG_C6_CME0_SHIFT              5
#define MCG_C6_CME0_WIDTH              1
#define MCG_C6_CME0(x) (((uint8_t)((uint8_t)(x)<<MCG_C6_CME0_SHIFT))&MCG_C6_CME0_MASK)

#define MCG_C6_PLLS_MASK                0x40u
#define MCG_C6_PLLS_SHIFT               6
#define MCG_C6_PLLS_WIDTH              1
#define MCG_C6_PLLS(x) (((uint8_t)((uint8_t)(x)<<MCG_C6_PLLS_SHIFT))&MCG_C6_PLLS_MASK)

#define MCG_C6_LOLIE0_MASK              0x80u
#define MCG_C6_LOLIE0_SHIFT             7
#define MCG_C6_LOLIE0_WIDTH             1
#define MCG_C6_LOLIE0(x) (((uint8_t)((uint8_t)(x)<<MCG_C6_LOLIE0_SHIFT))&MCG_C6_LOLIE0_MASK)

/* S Bit Fields */

#define MCG_S_IRCST_MASK                0x1u

```

```

#define MCG_S_IRCST_SHIFT          0
#define MCG_S_IRCST_WIDTH          1
#define MCG_S_IRCST(x) (((uint8_t)((uint8_t)(x)<<MCG_S_IRCST_SHIFT))&MCG_S_IRCST_MASK)

#define MCG_S_OSCINIT0_MASK         0x2u
#define MCG_S_OSCINIT0_SHIFT        1
#define MCG_S_OSCINIT0_WIDTH        1
#define MCG_S_OSCINIT0(x) (((uint8_t)((uint8_t)(x)<<MCG_S_OSCINIT0_SHIFT))&MCG_S_OSCINIT0_MASK)

#define MCG_S_CLKST_MASK            0xCu
#define MCG_S_CLKST_SHIFT           2
#define MCG_S_CLKST_WIDTH           2
#define MCG_S_CLKST(x) (((uint8_t)((uint8_t)(x)<<MCG_S_CLKST_SHIFT))&MCG_S_CLKST_MASK)

#define MCG_S_IREFST_MASK           0x10u
#define MCG_S_IREFST_SHIFT          4
#define MCG_S_IREFST_WIDTH          1
#define MCG_S_IREFST(x) (((uint8_t)((uint8_t)(x)<<MCG_S_IREFST_SHIFT))&MCG_S_IREFST_MASK)

#define MCG_S_PLLST_MASK             0x20u
#define MCG_S_PLLST_SHIFT            5
#define MCG_S_PLLST_WIDTH           1
#define MCG_S_PLLST(x) (((uint8_t)((uint8_t)(x)<<MCG_S_PLLST_SHIFT))&MCG_S_PLLST_MASK)

#define MCG_S_LOCK0_MASK             0x40u
#define MCG_S_LOCK0_SHIFT             6
#define MCG_S_LOCK0_WIDTH             1
#define MCG_S_LOCK0(x) (((uint8_t)((uint8_t)(x)<<MCG_S_LOCK0_SHIFT))&MCG_S_LOCK0_MASK)

#define MCG_S_LOLS0_MASK             0x80u
#define MCG_S_LOLS0_SHIFT              7

```

```

#define MCG_S_LOLS0_WIDTH           1
#define MCG_S_LOLS0(x)             (((uint8_t)((uint8_t)(x)<<MCG_S_LOLS0_SHIFT))&MCG_S_LOLS0_MASK)
/* SC Bit Fields */
#define MCG_SC_LOCS0_MASK          0x1u
#define MCG_SC_LOCS0_SHIFT         0
#define MCG_SC_LOCS0_WIDTH         1
#define MCG_SC_LOCS0(x)            (((uint8_t)((uint8_t)(x)<<MCG_SC_LOCS0_SHIFT))&MCG_SC_LOCS0_MASK)
#define MCG_SC_FCRDIV_MASK         0xEu
#define MCG_SC_FCRDIV_SHIFT        1
#define MCG_SC_FCRDIV_WIDTH        3
#define MCG_SC_FCRDIV(x)           (((uint8_t)((uint8_t)(x)<<MCG_SC_FCRDIV_SHIFT))&MCG_SC_FCRDIV_MASK)
#define MCG_SC_FLTPRSRV_MASK       0x10u
#define MCG_SC_FLTPRSRV_SHIFT      4
#define MCG_SC_FLTPRSRV_WIDTH      1
#define MCG_SC_FLTPRSRV(x)         (((uint8_t)((uint8_t)(x)<<MCG_SC_FLTPRSRV_SHIFT))&MCG_SC_FLTPRSRV_MASK)
#define MCG_SC_ATMF_MASK           0x20u
#define MCG_SC_ATMF_SHIFT          5
#define MCG_SC_ATMF_WIDTH          1
#define MCG_SC_ATMF(x)              (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMF_SHIFT))&MCG_SC_ATMF_MASK)
#define MCG_SC_ATMS_MASK           0x40u
#define MCG_SC_ATMS_SHIFT          6
#define MCG_SC_ATMS_WIDTH          1
#define MCG_SC_ATMS(x)              (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMS_SHIFT))&MCG_SC_ATMS_MASK)
#define MCG_SC_ATME_MASK           0x80u
#define MCG_SC_ATME_SHIFT          7

```

```

#define MCG_SC_ATME_WIDTH           1
#define MCG_SC_ATME(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATME_SHIFT))&MCG_SC_ATME_MASK)

/* ATCVH Bit Fields */

#define MCG_ATCVH_ATCVH_MASK        0xFFu
#define MCG_ATCVH_ATCVH_SHIFT       0
#define MCG_ATCVH_ATCVH_WIDTH       8
#define MCG_ATCVH_ATCVH(x) (((uint8_t)((uint8_t)(x)<<MCG_ATCVH_ATCVH_SHIFT))&MCG_ATCVH_ATCVH_MASK)

/* ATCVL Bit Fields */

#define MCG_ATCVL_ATCVL_MASK        0xFFu
#define MCG_ATCVL_ATCVL_SHIFT       0
#define MCG_ATCVL_ATCVL_WIDTH       8
#define MCG_ATCVL_ATCVL(x) (((uint8_t)((uint8_t)(x)<<MCG_ATCVL_ATCVL_SHIFT))&MCG_ATCVL_ATCVL_MASK)

/* C8 Bit Fields */

#define MCG_C8_LOLRE_MASK          0x40u
#define MCG_C8_LOLRE_SHIFT         6
#define MCG_C8_LOLRE_WIDTH         1
#define MCG_C8_LOLRE(x) (((uint8_t)((uint8_t)(x)<<MCG_C8_LOLRE_SHIFT))&MCG_C8_LOLRE_MASK)

/*!

 * @}
 */

/* /* end of group MCG_Register_Masks */

```

```

/* MCG - Peripheral instance base addresses */
/** Peripheral MCG base address */
#define MCG_BASE                   (0x40064000u)

```

```
/** Peripheral MCG base pointer */

#define MCG ((MCG_Type *)MCG_BASE)

#define MCG_BASE_PTR (MCG)

/** Array initializer of MCG peripheral base addresses */

#define MCG_BASE_ADDRS { MCG_BASE }

/** Array initializer of MCG peripheral base pointers */

#define MCG_BASE_PTRS { MCG }
```

```
/*
-----  
-- MCG - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros  
 * @{  
 */
```

```
/* MCG - Register instance definitions */

/* MCG */

#define MCG_C1 MCG_C1_REG(MCG)
#define MCG_C2 MCG_C2_REG(MCG)
#define MCG_C3 MCG_C3_REG(MCG)
#define MCG_C4 MCG_C4_REG(MCG)
#define MCG_C5 MCG_C5_REG(MCG)
#define MCG_C6 MCG_C6_REG(MCG)
#define MCG_S MCG_S_REG(MCG)
#define MCG_SC MCG_SC_REG(MCG)
#define MCG_ATCVH MCG_ATCVH_REG(MCG)
```

```
#define MCG_ATCVL          MCG_ATCVL_REG(MCG)
#define MCG_C7              MCG_C7_REG(MCG)
#define MCG_C8              MCG_C8_REG(MCG)
#define MCG_C9              MCG_C9_REG(MCG)
#define MCG_C10             MCG_C10_REG(MCG)
```

```
/*!
 * @}
 */
/* /* end of group MCG_Register_Accessor_Macros */
```

```
/* MCG C2[EREFS] backward compatibility */

#define MCG_C2_EREFs_MASK    (MCG_C2_EREFs0_MASK)
#define MCG_C2_EREFs_SHIFT   (MCG_C2_EREFs0_SHIFT)
#define MCG_C2_EREFs_WIDTH   (MCG_C2_EREFs0_WIDTH)
#define MCG_C2_EREFs(x)       (MCG_C2_EREFs0(x))
```

```
/* MCG C2[HGO] backward compatibility */

#define MCG_C2_HGO_MASK      (MCG_C2_HGO0_MASK)
#define MCG_C2_HGO_SHIFT     (MCG_C2_HGO0_SHIFT)
#define MCG_C2_HGO_WIDTH     (MCG_C2_HGO0_WIDTH)
#define MCG_C2_HGO(x)         (MCG_C2_HGO0(x))
```

```
/* MCG C2[RANGE] backward compatibility */

#define MCG_C2_RANGE_MASK    (MCG_C2_RANGE0_MASK)
#define MCG_C2_RANGE_SHIFT   (MCG_C2_RANGE0_SHIFT)
#define MCG_C2_RANGE_WIDTH   (MCG_C2_RANGE0_WIDTH)
#define MCG_C2_RANGE(x)       (MCG_C2_RANGE0(x))
```

```

/*!
* @}
* /* end of group MCG_Peripheral_Access_Layer */

/*
----- --
-- MCM Peripheral Access Layer
----- */
/*!
* @addtogroup MCM_Peripheral_Access_Layer MCM Peripheral Access Layer
* @{
*/
/** MCM - Register Layout Typedef */
typedef struct {
    uint8_t RESERVED_0[8];
    __I uint16_t PLASC;           /**< Crossbar Switch (AXBS) Slave Configuration, offset: 0x8 */
    __I uint16_t PLAMC;           /**< Crossbar Switch (AXBS) Master Configuration, offset: 0xA */
    __IO uint32_t PLACR;          /**< Platform Control Register, offset: 0xC */
    uint8_t RESERVED_1[48];
    __IO uint32_t CPO;            /**< Compute Operation Control Register, offset: 0x40 */
} MCM_Type, *MCM_MemMapPtr;

/*
----- --
-- MCM - Register accessor macros
----- */
/*!

```

```
* @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros
* @@
*/
```

```
/* MCM - Register accessors */

#define MCM_PLASC_REG(base) ((base)->PLASC)
#define MCM_PLAMC_REG(base) ((base)->PLAMC)
#define MCM_PLACR_REG(base) ((base)->PLACR)
#define MCM_CPO_REG(base) ((base)->CPO)
```

```
/*
* @@
*/
/* end of group MCM_Register_Accessor_Macros */
```

```
/*
-----  
-- MCM Register Masks  
----- */
```

```
/*
* @addtogroup MCM_Register_Masks MCM Register Masks
* @@
*/
```

```
/* PLASC Bit Fields */

#define MCM_PLASC_ASC_MASK 0xFFu
#define MCM_PLASC_ASC_SHIFT 0
#define MCM_PLASC_ASC_WIDTH 8
```

```

#define MCM_PLASC_ASC(x)
(((uint16_t)((uint16_t)(x)<<MCM_PLASC_ASC_SHIFT))&MCM_PLASC_ASC_MASK)

/* PLAMC Bit Fields */

#define MCM_PLAMC_AMC_MASK          0xFFu
#define MCM_PLAMC_AMC_SHIFT         0
#define MCM_PLAMC_AMC_WIDTH         8

#define MCM_PLAMC_AMC(x)
(((uint16_t)((uint16_t)(x)<<MCM_PLAMC_AMC_SHIFT))&MCM_PLAMC_AMC_MASK)

/* PLACR Bit Fields */

#define MCM_PLACR_ARB_MASK          0x200u
#define MCM_PLACR_ARB_SHIFT         9
#define MCM_PLACR_ARB_WIDTH         1

#define MCM_PLACR_ARB(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_ARB_SHIFT))&MCM_PLACR_ARB_MASK)

#define MCM_PLACR_CFCC_MASK          0x400u
#define MCM_PLACR_CFCC_SHIFT         10
#define MCM_PLACR_CFCC_WIDTH         1

#define MCM_PLACR_CFCC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_CFCC_SHIFT))&MCM_PLACR_CFCC_MASK)

#define MCM_PLACR_DFCDA_MASK          0x800u
#define MCM_PLACR_DFCDA_SHIFT         11
#define MCM_PLACR_DFCDA_WIDTH         1

#define MCM_PLACR_DFCDA(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCDA_SHIFT))&MCM_PLACR_DFCDA_MASK)

#define MCM_PLACR_DFCIC_MASK          0x1000u
#define MCM_PLACR_DFCIC_SHIFT         12
#define MCM_PLACR_DFCIC_WIDTH         1

#define MCM_PLACR_DFCIC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCIC_SHIFT))&MCM_PLACR_DFCIC_MASK)

#define MCM_PLACR_DFCC_MASK          0x2000u
#define MCM_PLACR_DFCC_SHIFT         13

```

```

#define MCM_PLACR_DFCC_WIDTH          1
#define MCM_PLACR_DFCC(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCC_SHIFT))&MCM_PLACR_DFCC_MASK)
#define MCM_PLACR_EFDS_MASK           0x4000u
#define MCM_PLACR_EFDS_SHIFT          14
#define MCM_PLACR_EFDS_WIDTH          1
#define MCM_PLACR_EFDS(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_EFDS_SHIFT))&MCM_PLACR_EFDS_MASK)
#define MCM_PLACR_DFCS_MASK           0x8000u
#define MCM_PLACR_DFCS_SHIFT          15
#define MCM_PLACR_DFCS_WIDTH          1
#define MCM_PLACR_DFCS(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCS_SHIFT))&MCM_PLACR_DFCS_MASK)
#define MCM_PLACR_ESFC_MASK           0x10000u
#define MCM_PLACR_ESFC_SHIFT          16
#define MCM_PLACR_ESFC_WIDTH          1
#define MCM_PLACR_ESFC(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_ESFC_SHIFT))&MCM_PLACR_ESFC_MASK)

/* CPO Bit Fields */
#define MCM_CPO_CPOREQ_MASK           0x1u
#define MCM_CPO_CPOREQ_SHIFT          0
#define MCM_CPO_CPOREQ_WIDTH          1
#define MCM_CPO_CPOREQ(x)             (((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOREQ_SHIFT))&MCM_CPO_CPOREQ_MASK)
#define MCM_CPO_CPOACK_MASK           0x2u
#define MCM_CPO_CPOACK_SHIFT          1
#define MCM_CPO_CPOACK_WIDTH          1
#define MCM_CPO_CPOACK(x)             (((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOACK_SHIFT))&MCM_CPO_CPOACK_MASK)
#define MCM_CPO_CPOWOI_MASK           0x4u
#define MCM_CPO_CPOWOI_SHIFT          2

```

```

#define MCM_CPO_CPOWOI_WIDTH           1

#define MCM_CPO_CPOWOI(x)
((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOWOI_SHIFT))&MCM_CPO_CPOWOI_MASK

/*
 * @}
 */

/* /* end of group MCM_Register_Masks */

/* MCM - Peripheral instance base addresses */
/** Peripheral MCM base address */
#define MCM_BASE                  (0xF0003000u)

/** Peripheral MCM base pointer */
#define MCM             ((MCM_Type *)MCM_BASE)

#define MCM_BASE_PTR            (MCM)

/** Array initializer of MCM peripheral base addresses */
#define MCM_BASE_ADDRS          { MCM_BASE }

/** Array initializer of MCM peripheral base pointers */
#define MCM_BASE_PTRS           { MCM }

/*
----- --
-- MCM - Register accessor macros
----- */

```

```

/*
 * @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros
 * @{
 */


```

```
/* MCM - Register instance definitions */

/* MCM */

#define MCM_PLASC          MCM_PLASC_REG(MCM)
#define MCM_PLAMC          MCM_PLAMC_REG(MCM)
#define MCM_PLACR          MCM_PLACR_REG(MCM)
#define MCM_CPO            MCM_CPO_REG(MCM)

/*!
 * @}

 */ /* end of group MCM_Register_Accessor_Macros */
```

```
/*!
 * @}

 */ /* end of group MCM_Peripheral_Access_Layer */
```

```
/* -----
-- MTB Peripheral Access Layer
----- */
```

```
/*!
 * @addtogroup MTB_Peripheral_Access_Layer MTB Peripheral Access Layer
 * @{
 *
 */
```

```
/** MTB - Register Layout Typedef */
```

```
typedef struct {
```

```

__IO uint32_t POSITION;           /**< MTB Position Register, offset: 0x0 */
__IO uint32_t MASTER;            /**< MTB Master Register, offset: 0x4 */
__IO uint32_t FLOW;              /**< MTB Flow Register, offset: 0x8 */
__I uint32_t BASE;               /**< MTB Base Register, offset: 0xC */

uint8_t RESERVED_0[3824];

__I uint32_t MODECTRL;          /**< Integration Mode Control Register, offset: 0xF00 */
uint8_t RESERVED_1[156];

__I uint32_t TAGSET;            /**< Claim TAG Set Register, offset: 0xFA0 */
__I uint32_t TAGCLEAR;          /**< Claim TAG Clear Register, offset: 0xFA4 */

uint8_t RESERVED_2[8];

__I uint32_t LOCKACCESS;         /**< Lock Access Register, offset: 0xFB0 */
__I uint32_t LOCKSTAT;          /**< Lock Status Register, offset: 0xFB4 */
__I uint32_t AUTHSTAT;          /**< Authentication Status Register, offset: 0xFB8 */
__I uint32_t DEVICEARCH;        /**< Device Architecture Register, offset: 0xFBC */

uint8_t RESERVED_3[8];

__I uint32_t DEVICECFG;         /**< Device Configuration Register, offset: 0xFC8 */
__I uint32_t DEVICETYPID;       /**< Device Type Identifier Register, offset: 0xFCC */
__I uint32_t PERIPHID[8];       /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4
*/
__I uint32_t COMPID[4];          /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
}

} MTB_Type, *MTB_MemMapPtr;

```

```

/*
-----  

-- MTB - Register accessor macros  

----- */

```

```
/*!
```

```
* @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros
```

```

* @@
*/
/* MTB - Register accessors */

#define MTB_POSITION_REG(base)      ((base)->POSITION)
#define MTB_MASTER_REG(base)        ((base)->MASTER)
#define MTB_FLOW_REG(base)          ((base)->FLOW)
#define MTB_BASE_REG(base)          ((base)->BASE)
#define MTB_MODECTRL_REG(base)      ((base)->MODECTRL)
#define MTB_TAGSET_REG(base)        ((base)->TAGSET)
#define MTB_TAGCLEAR_REG(base)      ((base)->TAGCLEAR)
#define MTB_LOCKACCESS_REG(base)    ((base)->LOCKACCESS)
#define MTB_LOCKSTAT_REG(base)      ((base)->LOCKSTAT)
#define MTB_AUTHSTAT_REG(base)      ((base)->AUTHSTAT)
#define MTB_DEVICEARCH_REG(base)    ((base)->DEVICEARCH)
#define MTB_DEVICECFG_REG(base)     ((base)->DEVICECFG)
#define MTB_DEVICETYPID_REG(base)   ((base)->DEVICETYPID)
#define MTB_PERIPHID_REG(base,index) ((base)->PERIPHID[index])
#define MTB_PERIPHID_COUNT          8
#define MTB_COMPID_REG(base,index)   ((base)->COMPID[index])
#define MTB_COMPID_COUNT            4

/*
* @@
*/
/* /* end of group MTB_Register_Accessor_Macros */

```

```

/* -----

```

```

-- MTB Register Masks
----- */

/*!
* @addtogroup MTB_Register_Masks MTB Register Masks
* @{
*/
/* POSITION Bit Fields */

#define MTB_POSITION_WRAP_MASK          0x4u
#define MTB_POSITION_WRAP_SHIFT         2
#define MTB_POSITION_WRAP_WIDTH         1
#define MTB_POSITION_WRAP(x)            (((uint32_t)((uint32_t)(x))<<MTB_POSITION_WRAP_SHIFT))&MTB_POSITION_WRAP_MASK

#define MTB_POSITION_POINTER_MASK        0xFFFFFFFF8u
#define MTB_POSITION_POINTER_SHIFT      3
#define MTB_POSITION_POINTER_WIDTH      29
#define MTB_POSITION_POINTER(x)         (((uint32_t)((uint32_t)(x))<<MTB_POSITION_POINTER_SHIFT))&MTB_POSITION_POINTER_MASK

/* MASTER Bit Fields */

#define MTB_MASTER_MASK_MASK           0x1Fu
#define MTB_MASTER_MASK_SHIFT          0
#define MTB_MASTER_MASK_WIDTH          5
#define MTB_MASTER_MASK(x)             (((uint32_t)((uint32_t)(x))<<MTB_MASTER_MASK_SHIFT))&MTB_MASTER_MASK_MASK

#define MTB_MASTER_TSTARTEN_MASK        0x20u
#define MTB_MASTER_TSTARTEN_SHIFT       5
#define MTB_MASTER_TSTARTEN_WIDTH       1
#define MTB_MASTER_TSTARTEN(x)          (((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTARTEN_SHIFT))&MTB_MASTER_TSTARTEN_MASK

```

```

#define MTB_MASTER_TSTOPEN_MASK      0x40u
#define MTB_MASTER_TSTOPEN_SHIFT     6
#define MTB_MASTER_TSTOPEN_WIDTH     1
#define MTB_MASTER_TSTOPEN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTOPEN_SHIFT))&MTB_MASTER_TSTOPEN_MASK)

#define MTB_MASTER_SFRWPRIV_MASK    0x80u
#define MTB_MASTER_SFRWPRIV_SHIFT   7
#define MTB_MASTER_SFRWPRIV_WIDTH   1
#define MTB_MASTER_SFRWPRIV(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_SFRWPRIV_SHIFT))&MTB_MASTER_SFRWPRIV_MASK)

#define MTB_MASTER_RAMPRIV_MASK     0x100u
#define MTB_MASTER_RAMPRIV_SHIFT    8
#define MTB_MASTER_RAMPRIV_WIDTH    1
#define MTB_MASTER_RAMPRIV(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_RAMPRIV_SHIFT))&MTB_MASTER_RAMPRIV_MASK)

#define MTB_MASTER_HALTREQ_MASK     0x200u
#define MTB_MASTER_HALTREQ_SHIFT    9
#define MTB_MASTER_HALTREQ_WIDTH    1
#define MTB_MASTER_HALTREQ(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_HALTREQ_SHIFT))&MTB_MASTER_HALTREQ_MASK)

#define MTB_MASTER_EN_MASK          0x80000000u
#define MTB_MASTER_EN_SHIFT         31
#define MTB_MASTER_EN_WIDTH         1
#define MTB_MASTER_EN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_EN_SHIFT))&MTB_MASTER_EN_MASK)

/* FLOW Bit Fields */

#define MTB_FLOW_AUTOSTOP_MASK      0x1u
#define MTB_FLOW_AUTOSTOP_SHIFT     0
#define MTB_FLOW_AUTOSTOP_WIDTH     1
#define MTB_FLOW_AUTOSTOP(x)
(((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOSTOP_SHIFT))&MTB_FLOW_AUTOSTOP_MASK)

```

```

#define MTB_FLOW_AUTOHALT_MASK      0x2u
#define MTB_FLOW_AUTOHALT_SHIFT     1
#define MTB_FLOW_AUTOHALT_WIDTH     1
#define MTB_FLOW_AUTOHALT(x) (((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOHALT_SHIFT))&MTB_FLOW_AUTOHALT_MASK)

#define MTB_FLOW_WATERMARK_MASK     0xFFFFFFFF8u
#define MTB_FLOW_WATERMARK_SHIFT    3
#define MTB_FLOW_WATERMARK_WIDTH    29
#define MTB_FLOW_WATERMARK(x) (((uint32_t)((uint32_t)(x))<<MTB_FLOW_WATERMARK_SHIFT))&MTB_FLOW_WATERMARK_MASK)

/* BASE Bit Fields */

#define MTB_BASE_BASEADDR_MASK      0xFFFFFFFFu
#define MTB_BASE_BASEADDR_SHIFT     0
#define MTB_BASE_BASEADDR_WIDTH     32
#define MTB_BASE_BASEADDR(x) (((uint32_t)((uint32_t)(x))<<MTB_BASE_BASEADDR_SHIFT))&MTB_BASE_BASEADDR_MASK)

/* MODECTRL Bit Fields */

#define MTB_MODECTRL_MODECTRL_MASK   0xFFFFFFFFu
#define MTB_MODECTRL_MODECTRL_SHIFT  0
#define MTB_MODECTRL_MODECTRL_WIDTH  32
#define MTB_MODECTRL_MODECTRL(x) (((uint32_t)((uint32_t)(x))<<MTB_MODECTRL_MODECTRL_SHIFT))&MTB_MODECTRL_MODECTRL_MASK)

/* TAGSET Bit Fields */

#define MTB_TAGSET_TAGSET_MASK      0xFFFFFFFFu
#define MTB_TAGSET_TAGSET_SHIFT     0
#define MTB_TAGSET_TAGSET_WIDTH     32
#define MTB_TAGSET_TAGSET(x) (((uint32_t)((uint32_t)(x))<<MTB_TAGSET_TAGSET_SHIFT))&MTB_TAGSET_TAGSET_MASK)

/* TAGCLEAR Bit Fields */

#define MTB_TAGCLEAR_TAGCLEAR_MASK  0xFFFFFFFFu

```

```

#define MTB_TAGCLEAR_TAGCLEAR_SHIFT      0
#define MTB_TAGCLEAR_TAGCLEAR_WIDTH     32
#define MTB_TAGCLEAR_TAGCLEAR(x)
(((uint32_t)((uint32_t)(x))<<MTB_TAGCLEAR_TAGCLEAR_SHIFT))&MTB_TAGCLEAR_TAGCLEAR_MASK)

/* LOCKACCESS Bit Fields */

#define MTB_LOCKACCESS_LOCKACCESS_MASK    0xFFFFFFFFu
#define MTB_LOCKACCESS_LOCKACCESS_SHIFT   0
#define MTB_LOCKACCESS_LOCKACCESS_WIDTH   32
#define MTB_LOCKACCESS_LOCKACCESS(x)
(((uint32_t)((uint32_t)(x))<<MTB_LOCKACCESS_LOCKACCESS_SHIFT))&MTB_LOCKACCESS_LOCKACCESS_MASK)

/* LOCKSTAT Bit Fields */

#define MTB_LOCKSTAT_LOCKSTAT_MASK       0xFFFFFFFFu
#define MTB_LOCKSTAT_LOCKSTAT_SHIFT     0
#define MTB_LOCKSTAT_LOCKSTAT_WIDTH     32
#define MTB_LOCKSTAT_LOCKSTAT(x)
(((uint32_t)((uint32_t)(x))<<MTB_LOCKSTAT_LOCKSTAT_SHIFT))&MTB_LOCKSTAT_LOCKSTAT_MASK)

/* AUTHSTAT Bit Fields */

#define MTB_AUTHSTAT_BIT0_MASK          0x1u
#define MTB_AUTHSTAT_BIT0_SHIFT         0
#define MTB_AUTHSTAT_BIT0_WIDTH         1
#define MTB_AUTHSTAT_BIT0(x)
(((uint32_t)((uint32_t)(x))<<MTB_AUTHSTAT_BIT0_SHIFT))&MTB_AUTHSTAT_BIT0_MASK)

#define MTB_AUTHSTAT_BIT1_MASK          0x2u
#define MTB_AUTHSTAT_BIT1_SHIFT         1
#define MTB_AUTHSTAT_BIT1_WIDTH         1
#define MTB_AUTHSTAT_BIT1(x)
(((uint32_t)((uint32_t)(x))<<MTB_AUTHSTAT_BIT1_SHIFT))&MTB_AUTHSTAT_BIT1_MASK)

#define MTB_AUTHSTAT_BIT2_MASK          0x4u
#define MTB_AUTHSTAT_BIT2_SHIFT         2
#define MTB_AUTHSTAT_BIT2_WIDTH         1

```

```

#define MTB_AUTHSTAT_BIT2(x)
(((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT2_SHIFT))&MTB_AUTHSTAT_BIT2_MASK)

#define MTB_AUTHSTAT_BIT3_MASK          0x8u
#define MTB_AUTHSTAT_BIT3_SHIFT        3
#define MTB_AUTHSTAT_BIT3_WIDTH        1

#define MTB_AUTHSTAT_BIT3(x)
(((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT3_SHIFT))&MTB_AUTHSTAT_BIT3_MASK)

/* DEVICEARCH Bit Fields */

#define MTB_DEVICEARCH_DEVICEARCH_MASK    0xFFFFFFFFu
#define MTB_DEVICEARCH_DEVICEARCH_SHIFT   0
#define MTB_DEVICEARCH_DEVICEARCH_WIDTH   32

#define MTB_DEVICEARCH_DEVICEARCH(x)
(((uint32_t)((uint32_t)(x)<<MTB_DEVICEARCH_DEVICEARCH_SHIFT))&MTB_DEVICEARCH_DEVICEARCH_MASK)

/* DEVICECFG Bit Fields */

#define MTB_DEVICECFG_DEVICECFG_MASK     0xFFFFFFFFu
#define MTB_DEVICECFG_DEVICECFG_SHIFT   0
#define MTB_DEVICECFG_DEVICECFG_WIDTH   32

#define MTB_DEVICECFG_DEVICECFG(x)
(((uint32_t)((uint32_t)(x)<<MTB_DEVICECFG_DEVICECFG_SHIFT))&MTB_DEVICECFG_DEVICECFG_MASK)

/* DEVICETYPID Bit Fields */

#define MTB_DEVICETYPID_DEVICETYPID_MASK 0xFFFFFFFFu
#define MTB_DEVICETYPID_DEVICETYPID_SHIFT 0
#define MTB_DEVICETYPID_DEVICETYPID_WIDTH 32

#define MTB_DEVICETYPID_DEVICETYPID(x)
(((uint32_t)((uint32_t)(x)<<MTB_DEVICETYPID_DEVICETYPID_SHIFT))&MTB_DEVICETYPID_DEVICETYPID_MASK)

/* PERIPHID Bit Fields */

#define MTB_PERIPHID_PERIPHID_MASK      0xFFFFFFFFu
#define MTB_PERIPHID_PERIPHID_SHIFT    0
#define MTB_PERIPHID_PERIPHID_WIDTH    32

```

```

#define MTB_PERIPHID_PERIPHID(x)
(((uint32_t)((uint32_t)(x))<<MTB_PERIPHID_PERIPHID_SHIFT))&MTB_PERIPHID_PERIPHID_MASK)

/* COMPID Bit Fields */

#define MTB_COMPID_COMPID_MASK           0xFFFFFFFFu
#define MTB_COMPID_COMPID_SHIFT          0
#define MTB_COMPID_COMPID_WIDTH          32

#define MTB_COMPID_COMPID(x)
(((uint32_t)((uint32_t)(x))<<MTB_COMPID_COMPID_SHIFT))&MTB_COMPID_COMPID_MASK)

/*
 * @}
 */

/* /* end of group MTB_Register_Masks */

```

```

/* MTB - Peripheral instance base addresses */

/** Peripheral MTB base address */

#define MTB_BASE             (0xF0000000u)

/** Peripheral MTB base pointer */

#define MTB                 ((MTB_Type *)MTB_BASE)

#define MTB_BASE_PTR         (MTB)

/** Array initializer of MTB peripheral base addresses */

#define MTB_BASE_ADDRS       { MTB_BASE }

/** Array initializer of MTB peripheral base pointers */

#define MTB_BASE_PTRS        { MTB }

```

```

/*
-----  

-- MTB - Register accessor macros  

----- */

```

```
/*!  
 * @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros  
 * @{  
 */
```

```
/* MTB - Register instance definitions */  
  
/* MTB */  
  
#define MTB_POSITION           MTB_POSITION_REG(MTB)  
#define MTB_MASTER              MTB_MASTER_REG(MTB)  
#define MTB_FLOW                MTB_FLOW_REG(MTB)  
#define MTB_BASEr               MTB_BASE_REG(MTB)  
#define MTB_MODECTRL            MTB_MODECTRL_REG(MTB)  
#define MTB_TAGSET              MTB_TAGSET_REG(MTB)  
#define MTB_TAGCLEAR             MTB_TAGCLEAR_REG(MTB)  
#define MTB_LOCKACCESS           MTB_LOCKACCESS_REG(MTB)  
#define MTB_LOCKSTAT             MTB_LOCKSTAT_REG(MTB)  
#define MTB_AUTHSTAT             MTB_AUTHSTAT_REG(MTB)  
#define MTB_DEVICESEARCH          MTB_DEVICESEARCH_REG(MTB)  
#define MTB_DEVICECFG             MTB_DEVICECFG_REG(MTB)  
#define MTB_DEVICETYPID          MTB_DEVICETYPID_REG(MTB)  
#define MTB_PERIPHID4            MTB_PERIPHID_REG(MTB,0)  
#define MTB_PERIPHID5            MTB_PERIPHID_REG(MTB,1)  
#define MTB_PERIPHID6            MTB_PERIPHID_REG(MTB,2)  
#define MTB_PERIPHID7            MTB_PERIPHID_REG(MTB,3)  
#define MTB_PERIPHID0            MTB_PERIPHID_REG(MTB,4)  
#define MTB_PERIPHID1            MTB_PERIPHID_REG(MTB,5)  
#define MTB_PERIPHID2            MTB_PERIPHID_REG(MTB,6)  
#define MTB_PERIPHID3            MTB_PERIPHID_REG(MTB,7)
```

```
#define MTB_COMPID0          MTB_COMPID_REG(MTB,0)
#define MTB_COMPID1          MTB_COMPID_REG(MTB,1)
#define MTB_COMPID2          MTB_COMPID_REG(MTB,2)
#define MTB_COMPID3          MTB_COMPID_REG(MTB,3)

/* MTB - Register array accessors */

#define MTB_PERIPHID(index)    MTB_PERIPHID_REG(MTB,index)
#define MTB_COMPID(index)       MTB_COMPID_REG(MTB,index)
```

```
/*
 * @}
 */
/* end of group MTB_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group MTB_Peripheral_Access_Layer */
```

```
/*
-----  
-- MTBDWT Peripheral Access Layer  
----- */
```

```
/*
 * @addtogroup MTBDWT_Peripheral_Access_Layer MTBDWT Peripheral Access Layer
 * @{
 */

```

```
/** MTBDWT - Register Layout Typedef */
```

```

typedef struct {

    __I uint32_t CTRL;           /**< MTB DWT Control Register, offset: 0x0 */

    uint8_t RESERVED_0[28];

    struct {                   /* offset: 0x20, array step: 0x10 */

        __IO uint32_t COMP;      /**< MTB_DWT Comparator Register, array offset: 0x20, array
step: 0x10 */

        __IO uint32_t MASK;     /**< MTB_DWT Comparator Mask Register, array offset: 0x24,
array step: 0x10 */

        __IO uint32_t FCT;      /**< MTB_DWT Comparator Function Register 0..MTB_DWT
Comparator Function Register 1, array offset: 0x28, array step: 0x10 */

        uint8_t RESERVED_0[4];
    } COMPARATOR[2];

    uint8_t RESERVED_1[448];

    __IO uint32_t TBCTRL;       /**< MTB_DWT Trace Buffer Control Register, offset: 0x200 */

    uint8_t RESERVED_2[3524];

    __I uint32_t DEVICECFG;    /**< Device Configuration Register, offset: 0xFC8 */

    __I uint32_t DEVICETYPID;  /**< Device Type Identifier Register, offset: 0xFCC */

    __I uint32_t PERIPHID[8];  /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4
*/
}

__I uint32_t COMPID[4];      /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
} MTBDWT_Type, *MTBDWT_MemMapPtr;

```

```

/*
-----*
-- MTBDWT - Register accessor macros
----- */

```

```

/*!
* @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros
* @{

```

```
*/
```

```
/* MTBDWT - Register accessors */

#define MTBDWT_CTRL_REG(base)          ((base)->CTRL)
#define MTBDWT_COMP_REG(base,index)     ((base)->COMPARATOR[index].COMP)
#define MTBDWT_COMP_COUNT             2
#define MTBDWT_MASK_REG(base,index)    ((base)->COMPARATOR[index].MASK)
#define MTBDWT_MASK_COUNT             2
#define MTBDWT_FCT_REG(base,index)     ((base)->COMPARATOR[index].FCT)
#define MTBDWT_FCT_COUNT              2
#define MTBDWT_TBCTRL_REG(base)        ((base)->TBCTRL)
#define MTBDWT_DEVICECFG_REG(base)     ((base)->DEVICECFG)
#define MTBDWT_DEVICETYPID_REG(base)   ((base)->DEVICETYPID)
#define MTBDWT_PERIPHID_REG(base,index) ((base)->PERIPHID[index])
#define MTBDWT_PERIPHID_COUNT         8
#define MTBDWT_COMPID_REG(base,index)  ((base)->COMPID[index])
#define MTBDWT_COMPID_COUNT            4

/*!
 * @}
 */
/* /* end of group MTBDWT_Register_Accessor_Macros */
```

```
/*
-----  
-- MTBDWT Register Masks  
----- */
```

```
/*!
```

```

* @addtogroup MTBDWT_Register_Masks MTBDWT Register Masks
* @{
* /

/* CTRL Bit Fields */

#define MTBDWT_CTRL_DWTCFGCTRL_MASK      0xFFFFFFFFu
#define MTBDWT_CTRL_DWTCFGCTRL_SHIFT     0
#define MTBDWT_CTRL_DWTCFGCTRL_WIDTH     28
#define MTBDWT_CTRL_DWTCFGCTRL(x)        (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_DWTCFGCTRL_SHIFT))&MTBDWT_CTRL_DWTCFGCTRL_MASK)
#define MTBDWT_CTRL_NUMCMP_MASK          0xF0000000u
#define MTBDWT_CTRL_NUMCMP_SHIFT         28
#define MTBDWT_CTRL_NUMCMP_WIDTH         4
#define MTBDWT_CTRL_NUMCMP(x)           (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_NUMCMP_SHIFT))&MTBDWT_CTRL_NUMCMP_MASK)

/* COMP Bit Fields */

#define MTBDWT_COMP_COMP_MASK           0xFFFFFFFFu
#define MTBDWT_COMP_COMP_SHIFT          0
#define MTBDWT_COMP_COMP_WIDTH          32
#define MTBDWT_COMP_COMP(x)             (((uint32_t)((uint32_t)(x))<<MTBDWT_COMP_COMP_SHIFT))&MTBDWT_COMP_COMP_MASK)

/* MASK Bit Fields */

#define MTBDWT_MASK_MASK_MASK          0x1Fu
#define MTBDWT_MASK_MASK_SHIFT          0
#define MTBDWT_MASK_MASK_WIDTH          5
#define MTBDWT_MASK_MASK(x)            (((uint32_t)((uint32_t)(x))<<MTBDWT_MASK_MASK_SHIFT))&MTBDWT_MASK_MASK_MASK)

/* FCT Bit Fields */

#define MTBDWT_FCT_FUNCTION_MASK       0xFu

```

```

#define MTBDWT_FCT_FUNCTION_SHIFT      0
#define MTBDWT_FCT_FUNCTION_WIDTH     4
#define MTBDWT_FCT_FUNCTION(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_FUNCTION_SHIFT))&MTBDWT_FCT_FUNCTION_MASK)

#define MTBDWT_FCT_DATAVMATCH_MASK    0x100u
#define MTBDWT_FCT_DATAVMATCH_SHIFT   8
#define MTBDWT_FCT_DATAVMATCH_WIDTH   1
#define MTBDWT_FCT_DATAVMATCH(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVMATCH_SHIFT))&MTBDWT_FCT_DATAVMATCH_MASK)

#define MTBDWT_FCT_DATAVSIZE_MASK     0xC00u
#define MTBDWT_FCT_DATAVSIZE_SHIFT    10
#define MTBDWT_FCT_DATAVSIZE_WIDTH    2
#define MTBDWT_FCT_DATAVSIZE(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVSIZE_SHIFT))&MTBDWT_FCT_DATAVSIZE_MASK)

#define MTBDWT_FCT_DATAVADDR0_MASK    0xF000u
#define MTBDWT_FCT_DATAVADDR0_SHIFT   12
#define MTBDWT_FCT_DATAVADDR0_WIDTH   4
#define MTBDWT_FCT_DATAVADDR0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVADDR0_SHIFT))&MTBDWT_FCT_DATAVADDR0_MASK)

#define MTBDWT_FCT_MATCHED_MASK       0x1000000u
#define MTBDWT_FCT_MATCHED_SHIFT      24
#define MTBDWT_FCT_MATCHED_WIDTH     1
#define MTBDWT_FCT_MATCHED(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_MATCHED_SHIFT))&MTBDWT_FCT_MATCHED_MASK)

/* TBCTRL Bit Fields */

#define MTBDWT_TBCTRL_ACOMP0_MASK     0x1u
#define MTBDWT_TBCTRL_ACOMP0_SHIFT    0
#define MTBDWT_TBCTRL_ACOMP0_WIDTH    1
#define MTBDWT_TBCTRL_ACOMP0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP0_SHIFT))&MTBDWT_TBCTRL_ACOMP0_MASK)

```

```

#define MTBDWT_TBCTRL_ACOMP1_MASK      0x2u
#define MTBDWT_TBCTRL_ACOMP1_SHIFT     1
#define MTBDWT_TBCTRL_ACOMP1_WIDTH     1
#define MTBDWT_TBCTRL_ACOMP1(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP1_SHIFT))&MTBDWT_TBCTRL_ACOMP1_MASK)

#define MTBDWT_TBCTRL_NUMCOMP_MASK    0xF0000000u
#define MTBDWT_TBCTRL_NUMCOMP_SHIFT   28
#define MTBDWT_TBCTRL_NUMCOMP_WIDTH   4
#define MTBDWT_TBCTRL_NUMCOMP(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_NUMCOMP_SHIFT))&MTBDWT_TBCTRL_NUMCOMP_M
ASK)

/* DEVICECFG Bit Fields */

#define MTBDWT_DEVICECFG_DEVICECFG_MASK 0xFFFFFFFFu
#define MTBDWT_DEVICECFG_DEVICECFG_SHIFT 0
#define MTBDWT_DEVICECFG_DEVICECFG_WIDTH 32
#define MTBDWT_DEVICECFG_DEVICECFG(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICECFG_DEVICECFG_SHIFT))&MTBDWT_DEVICECFG_DEVICE
CFG_MASK)

/* DEVICETYPID Bit Fields */

#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK 0xFFFFFFFFu
#define MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT 0
#define MTBDWT_DEVICETYPID_DEVICETYPID_WIDTH 32
#define MTBDWT_DEVICETYPID_DEVICETYPID(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT))&MTBDWT_DEVICETYPID_DE
VICETYPID_MASK)

/* PERIPHID Bit Fields */

#define MTBDWT_PERIPHID_PERIPHID_MASK 0xFFFFFFFFu
#define MTBDWT_PERIPHID_PERIPHID_SHIFT 0
#define MTBDWT_PERIPHID_PERIPHID_WIDTH 32
#define MTBDWT_PERIPHID_PERIPHID(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_PERIPHID_PERIPHID_SHIFT))&MTBDWT_PERIPHID_PERIPHID_M
ASK)

```

```
/* COMPID Bit Fields */

#define MTBDWT_COMPID_COMPID_MASK      0xFFFFFFFFu
#define MTBDWT_COMPID_COMPID_SHIFT     0
#define MTBDWT_COMPID_COMPID_WIDTH     32
#define MTBDWT_COMPID_COMPID(x)        (((uint32_t)((uint32_t)(x)<<MTBDWT_COMPID_COMPID_SHIFT))&MTBDWT_COMPID_COMPID_MASK)
```

```
/*
 * @}
 */
/* /* end of group MTBDWT_Register_Masks */
```

```
/* MTBDWT - Peripheral instance base addresses */

/** Peripheral MTBDWT base address */

#define MTBDWT_BASE          (0xF0001000u)

/** Peripheral MTBDWT base pointer */

#define MTBDWT                ((MTBDWT_Type *)MTBDWT_BASE)
#define MTBDWT_BASE_PTR        (MTBDWT)

/** Array initializer of MTBDWT peripheral base addresses */

#define MTBDWT_BASE_ADDRS     { MTBDWT_BASE }

/** Array initializer of MTBDWT peripheral base pointers */

#define MTBDWT_BASE_PTRS      { MTBDWT }
```

```
/*
-- MTBDWT - Register accessor macros
----- */
```

```
/*
 * @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros
```

```

* @{

*/



/* MTBDWT - Register instance definitions */

/* MTBDWT */

#define MTBDWT_CTRL           MTBDWT_CTRL_REG(MTBDWT)
#define MTBDWT_COMP0          MTBDWT_COMP_REG(MTBDWT,0)
#define MTBDWT_MASK0          MTBDWT_MASK_REG(MTBDWT,0)
#define MTBDWT_FCT0            MTBDWT_FCT_REG(MTBDWT,0)
#define MTBDWT_COMP1          MTBDWT_COMP_REG(MTBDWT,1)
#define MTBDWT_MASK1          MTBDWT_MASK_REG(MTBDWT,1)
#define MTBDWT_FCT1            MTBDWT_FCT_REG(MTBDWT,1)
#define MTBDWT_TBCTRL          MTBDWT_TBCTRL_REG(MTBDWT)
#define MTBDWT_DEVICECFG        MTBDWT_DEVICECFG_REG(MTBDWT)
#define MTBDWT_DEVICETYPID      MTBDWT_DEVICETYPID_REG(MTBDWT)
#define MTBDWT_PERIPHID4        MTBDWT_PERIPHID_REG(MTBDWT,0)
#define MTBDWT_PERIPHID5        MTBDWT_PERIPHID_REG(MTBDWT,1)
#define MTBDWT_PERIPHID6        MTBDWT_PERIPHID_REG(MTBDWT,2)
#define MTBDWT_PERIPHID7        MTBDWT_PERIPHID_REG(MTBDWT,3)
#define MTBDWT_PERIPHID0        MTBDWT_PERIPHID_REG(MTBDWT,4)
#define MTBDWT_PERIPHID1        MTBDWT_PERIPHID_REG(MTBDWT,5)
#define MTBDWT_PERIPHID2        MTBDWT_PERIPHID_REG(MTBDWT,6)
#define MTBDWT_PERIPHID3        MTBDWT_PERIPHID_REG(MTBDWT,7)
#define MTBDWT_COMPID0          MTBDWT_COMPID_REG(MTBDWT,0)
#define MTBDWT_COMPID1          MTBDWT_COMPID_REG(MTBDWT,1)
#define MTBDWT_COMPID2          MTBDWT_COMPID_REG(MTBDWT,2)
#define MTBDWT_COMPID3          MTBDWT_COMPID_REG(MTBDWT,3)

```

```

/* MTBDWT - Register array accessors */

#define MTBDWT_COMP(index)          MTBDWT_COMP_REG(MTBDWT,index)
#define MTBDWT_MASK(index)          MTBDWT_MASK_REG(MTBDWT,index)
#define MTBDWT_FCT(index)           MTBDWT_FCT_REG(MTBDWT,index)
#define MTBDWT_PERIPHID(index)      MTBDWT_PERIPHID_REG(MTBDWT,index)
#define MTBDWT_COMPID(index)         MTBDWT_COMPID_REG(MTBDWT,index)

/*!
 * @}
 */
/* /* end of group MTBDWT_Register_Accessor_Macros */

```

```

/*!
 * @}
 */
/* /* end of group MTBDWT_Peripheral_Access_Layer */

```

```

/*
-----  

-- NV Peripheral Access Layer  

----- */

```

```

/*!  

 * @addtogroup NV_Peripheral_Access_Layer NV Peripheral Access Layer  

 * @{
 * /  


```

```

/** NV - Register Layout Typedef */  

typedef struct {  

    __I uint8_t BACKKEY3;           /**< Backdoor Comparison Key 3., offset: 0x0 */

```

```

__I uint8_t BACKKEY2;           /**< Backdoor Comparison Key 2., offset: 0x1 */
__I uint8_t BACKKEY1;           /**< Backdoor Comparison Key 1., offset: 0x2 */
__I uint8_t BACKKEY0;           /**< Backdoor Comparison Key 0., offset: 0x3 */
__I uint8_t BACKKEY7;           /**< Backdoor Comparison Key 7., offset: 0x4 */
__I uint8_t BACKKEY6;           /**< Backdoor Comparison Key 6., offset: 0x5 */
__I uint8_t BACKKEY5;           /**< Backdoor Comparison Key 5., offset: 0x6 */
__I uint8_t BACKKEY4;           /**< Backdoor Comparison Key 4., offset: 0x7 */
__I uint8_t FPROT3;             /**< Non-volatile P-Flash Protection 1 - Low Register, offset: 0x8 */
*/
__I uint8_t FPROT2;             /**< Non-volatile P-Flash Protection 1 - High Register, offset: 0x9 */
*/
__I uint8_t FPROT1;             /**< Non-volatile P-Flash Protection 0 - Low Register, offset: 0xA */
*/
__I uint8_t FPROT0;             /**< Non-volatile P-Flash Protection 0 - High Register, offset: 0xB */
*/
__I uint8_t FSEC;               /**< Non-volatile Flash Security Register, offset: 0xC */
__I uint8_t FOPT;               /**< Non-volatile Flash Option Register, offset: 0xD */

} NV_Type, *NV_MemMapPtr;

```

```

/*
-----  

-- NV - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros  

* @{
*/

```

```

/* NV - Register accessors */

```

```
#define NV_BACKKEY3_REG(base)          ((base)->BACKKEY3)
#define NV_BACKKEY2_REG(base)          ((base)->BACKKEY2)
#define NV_BACKKEY1_REG(base)          ((base)->BACKKEY1)
#define NV_BACKKEY0_REG(base)          ((base)->BACKKEY0)
#define NV_BACKKEY7_REG(base)          ((base)->BACKKEY7)
#define NV_BACKKEY6_REG(base)          ((base)->BACKKEY6)
#define NV_BACKKEY5_REG(base)          ((base)->BACKKEY5)
#define NV_BACKKEY4_REG(base)          ((base)->BACKKEY4)
#define NV_FPROT3_REG(base)           ((base)->FPROT3)
#define NV_FPROT2_REG(base)           ((base)->FPROT2)
#define NV_FPROT1_REG(base)           ((base)->FPROT1)
#define NV_FPROT0_REG(base)           ((base)->FPROT0)
#define NV_FSEC_REG(base)             ((base)->FSEC)
#define NV_FOPT_REG(base)             ((base)->FOPT)
```

```
/*
 * @}
 */
/* /* end of group NV_Register_Accessor_Macros */
```

```
/*
-----  
-- NV Register Masks  
----- */
```

```
/*
 * @addtogroup NV_Register_Masks NV Register Masks
 * @{
 */
*/
```

```

/* BACKKEY3 Bit Fields */

#define NV_BACKKEY3_KEY_MASK      0xFFu
#define NV_BACKKEY3_KEY_SHIFT     0
#define NV_BACKKEY3_KEY_WIDTH     8
#define NV_BACKKEY3_KEY(x)        (((uint8_t)((uint8_t)(x)<<NV_BACKKEY3_KEY_SHIFT))&NV_BACKKEY3_KEY_MASK)

/* BACKKEY2 Bit Fields */

#define NV_BACKKEY2_KEY_MASK      0xFFu
#define NV_BACKKEY2_KEY_SHIFT     0
#define NV_BACKKEY2_KEY_WIDTH     8
#define NV_BACKKEY2_KEY(x)        (((uint8_t)((uint8_t)(x)<<NV_BACKKEY2_KEY_SHIFT))&NV_BACKKEY2_KEY_MASK)

/* BACKKEY1 Bit Fields */

#define NV_BACKKEY1_KEY_MASK      0xFFu
#define NV_BACKKEY1_KEY_SHIFT     0
#define NV_BACKKEY1_KEY_WIDTH     8
#define NV_BACKKEY1_KEY(x)        (((uint8_t)((uint8_t)(x)<<NV_BACKKEY1_KEY_SHIFT))&NV_BACKKEY1_KEY_MASK)

/* BACKKEY0 Bit Fields */

#define NV_BACKKEY0_KEY_MASK      0xFFu
#define NV_BACKKEY0_KEY_SHIFT     0
#define NV_BACKKEY0_KEY_WIDTH     8
#define NV_BACKKEY0_KEY(x)        (((uint8_t)((uint8_t)(x)<<NV_BACKKEY0_KEY_SHIFT))&NV_BACKKEY0_KEY_MASK)

/* BACKKEY7 Bit Fields */

#define NV_BACKKEY7_KEY_MASK      0xFFu
#define NV_BACKKEY7_KEY_SHIFT     0
#define NV_BACKKEY7_KEY_WIDTH     8
#define NV_BACKKEY7_KEY(x)        (((uint8_t)((uint8_t)(x)<<NV_BACKKEY7_KEY_SHIFT))&NV_BACKKEY7_KEY_MASK)

```

```

/* BACKKEY6 Bit Fields */

#define NV_BACKKEY6_KEY_MASK          0xFFu
#define NV_BACKKEY6_KEY_SHIFT         0
#define NV_BACKKEY6_KEY_WIDTH         8
#define NV_BACKKEY6_KEY(x)            (((uint8_t)((uint8_t)(x)<<NV_BACKKEY6_KEY_SHIFT))&NV_BACKKEY6_KEY_MASK)

/* BACKKEY5 Bit Fields */

#define NV_BACKKEY5_KEY_MASK          0xFFu
#define NV_BACKKEY5_KEY_SHIFT         0
#define NV_BACKKEY5_KEY_WIDTH         8
#define NV_BACKKEY5_KEY(x)            (((uint8_t)((uint8_t)(x)<<NV_BACKKEY5_KEY_SHIFT))&NV_BACKKEY5_KEY_MASK)

/* BACKKEY4 Bit Fields */

#define NV_BACKKEY4_KEY_MASK          0xFFu
#define NV_BACKKEY4_KEY_SHIFT         0
#define NV_BACKKEY4_KEY_WIDTH         8
#define NV_BACKKEY4_KEY(x)            (((uint8_t)((uint8_t)(x)<<NV_BACKKEY4_KEY_SHIFT))&NV_BACKKEY4_KEY_MASK)

/* FPROT3 Bit Fields */

#define NV_FPROT3_PROT_MASK           0xFFu
#define NV_FPROT3_PROT_SHIFT          0
#define NV_FPROT3_PROT_WIDTH          8
#define NV_FPROT3_PROT(x)             (((uint8_t)((uint8_t)(x)<<NV_FPROT3_PROT_SHIFT))&NV_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

#define NV_FPROT2_PROT_MASK           0xFFu
#define NV_FPROT2_PROT_SHIFT          0
#define NV_FPROT2_PROT_WIDTH          8
#define NV_FPROT2_PROT(x)             (((uint8_t)((uint8_t)(x)<<NV_FPROT2_PROT_SHIFT))&NV_FPROT2_PROT_MASK)

```

```

/* FPROT1 Bit Fields */

#define NV_FPROT1_PROT_MASK          0xFFu
#define NV_FPROT1_PROT_SHIFT         0
#define NV_FPROT1_PROT_WIDTH         8
#define NV_FPROT1_PROT(x)            (((uint8_t)((uint8_t)(x)<<NV_FPROT1_PROT_SHIFT))&NV_FPROT1_PROT_MASK)

/* FPROTO Bit Fields */

#define NV_FPROTO_PROT_MASK          0xFFu
#define NV_FPROTO_PROT_SHIFT         0
#define NV_FPROTO_PROT_WIDTH         8
#define NV_FPROTO_PROT(x)            (((uint8_t)((uint8_t)(x)<<NV_FPROTO_PROT_SHIFT))&NV_FPROTO_PROT_MASK)

/* FSEC Bit Fields */

#define NV_FSEC_SEC_MASK             0x3u
#define NV_FSEC_SEC_SHIFT            0
#define NV_FSEC_SEC_WIDTH            2
#define NV_FSEC_SEC(x)               (((uint8_t)((uint8_t)(x)<<NV_FSEC_SEC_SHIFT))&NV_FSEC_SEC_MASK)

#define NV_FSEC_FSLACC_MASK          0xCu
#define NV_FSEC_FSLACC_SHIFT         2
#define NV_FSEC_FSLACC_WIDTH         2
#define NV_FSEC_FSLACC(x)            (((uint8_t)((uint8_t)(x)<<NV_FSEC_FSLACC_SHIFT))&NV_FSEC_FSLACC_MASK)

#define NV_FSEC_MEEN_MASK            0x30u
#define NV_FSEC_MEEN_SHIFT           4
#define NV_FSEC_MEEN_WIDTH           2
#define NV_FSEC_MEEN(x)              (((uint8_t)((uint8_t)(x)<<NV_FSEC_MEEN_SHIFT))&NV_FSEC_MEEN_MASK)

#define NV_FSEC_KEYEN_MASK           0xC0u
#define NV_FSEC_KEYEN_SHIFT          6

```

```

#define NV_FSEC_KEYEN_WIDTH          2
#define NV_FSEC_KEYEN(x) (((uint8_t)((uint8_t)(x)<<NV_FSEC_KEYEN_SHIFT))&NV_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define NV_FOPT_LPBOOT0_MASK         0x1u
#define NV_FOPT_LPBOOT0_SHIFT        0
#define NV_FOPT_LPBOOT0_WIDTH        1
#define NV_FOPT_LPBOOT0(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT0_SHIFT))&NV_FOPT_LPBOOT0_MASK)

#define NV_FOPT_NMI_DIS_MASK         0x4u
#define NV_FOPT_NMI_DIS_SHIFT        2
#define NV_FOPT_NMI_DIS_WIDTH        1
#define NV_FOPT_NMI_DIS(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_NMI_DIS_SHIFT))&NV_FOPT_NMI_DIS_MASK)

#define NV_FOPT_RESET_PIN_CFG_MASK   0x8u
#define NV_FOPT_RESET_PIN_CFG_SHIFT  3
#define NV_FOPT_RESET_PIN_CFG_WIDTH  1
#define NV_FOPT_RESET_PIN_CFG(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_RESET_PIN_CFG_SHIFT))&NV_FOPT_RESET_PIN_CFG_MASK)

#define NV_FOPT_LPBOOT1_MASK         0x10u
#define NV_FOPT_LPBOOT1_SHIFT        4
#define NV_FOPT_LPBOOT1_WIDTH        1
#define NV_FOPT_LPBOOT1(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT1_SHIFT))&NV_FOPT_LPBOOT1_MASK)

#define NV_FOPT_FAST_INIT_MASK       0x20u
#define NV_FOPT_FAST_INIT_SHIFT      5
#define NV_FOPT_FAST_INIT_WIDTH      1
#define NV_FOPT_FAST_INIT(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_FAST_INIT_SHIFT))&NV_FOPT_FAST_INIT_MASK)

/*

```

```

* @}

/* /* end of group NV_Register_Masks */

/* NV - Peripheral instance base addresses */

/** Peripheral FTFA_FlashConfig base address */

#define FTFA_FlashConfig_BASE      (0x400u)

/** Peripheral FTFA_FlashConfig base pointer */

#define FTFA_FlashConfig          ((NV_Type *)FTFA_FlashConfig_BASE)

#define FTFA_FlashConfig_BASE_PTR  (FTFA_FlashConfig)

/** Array initializer of NV peripheral base addresses */

#define NV_BASE_ADDRS            { FTFA_FlashConfig_BASE }

/** Array initializer of NV peripheral base pointers */

#define NV_BASE_PTRS              { FTFA_FlashConfig }

```

```

/*
-----  

-- NV - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros  

* @{
* /

```

```

/* NV - Register instance definitions */

/* FTFA_FlashConfig */

#define NV_BACKKEY3           NV_BACKKEY3_REG(FTFA_FlashConfig)  

#define NV_BACKKEY2           NV_BACKKEY2_REG(FTFA_FlashConfig)

```

```
#define NV_BACKKEY1           NV_BACKKEY1_REG(FTFA_FlashConfig)
#define NV_BACKKEY0           NV_BACKKEY0_REG(FTFA_FlashConfig)
#define NV_BACKKEY7           NV_BACKKEY7_REG(FTFA_FlashConfig)
#define NV_BACKKEY6           NV_BACKKEY6_REG(FTFA_FlashConfig)
#define NV_BACKKEY5           NV_BACKKEY5_REG(FTFA_FlashConfig)
#define NV_BACKKEY4           NV_BACKKEY4_REG(FTFA_FlashConfig)
#define NV_FPROT3             NV_FPROT3_REG(FTFA_FlashConfig)
#define NV_FPROT2             NV_FPROT2_REG(FTFA_FlashConfig)
#define NV_FPROT1             NV_FPROT1_REG(FTFA_FlashConfig)
#define NV_FPROT0             NV_FPROT0_REG(FTFA_FlashConfig)
#define NV_FSEC               NV_FSEC_REG(FTFA_FlashConfig)
#define NV_FOPT               NV_FOPT_REG(FTFA_FlashConfig)
```

```
/*!
 * @}
 */
/* end of group NV_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* end of group NV_Peripheral_Access_Layer */
```

```
/*
-----  
-- OSC Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup OSC_Peripheral_Access_Layer OSC Peripheral Access Layer
```

```
* @{

*/

```

```
/** OSC - Register Layout Typedef */

typedef struct {

    __IO uint8_t CR;           /* < OSC Control Register, offset: 0x0 */

} OSC_Type, *OSC_MemMapPtr;

```

```
/*
-----  
-- OSC - Register accessor macros  
----- */

```

```
/*!  
* @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros  
* @{
*/

```

```
/* OSC - Register accessors */  
#define OSC_CR_REG(base) ((base)->CR)
```

```
/*!  
* @}  
* /* end of group OSC_Register_Accessor_Macros */

```

```
/*
-----  
-- OSC Register Masks  
----- */

```

```

/*!

* @addtogroup OSC_Register_Masks OSC Register Masks

* @{

*/



/* CR Bit Fields */

#define OSC_CR_SC16P_MASK          0x1u
#define OSC_CR_SC16P_SHIFT         0
#define OSC_CR_SC16P_WIDTH          1
#define OSC_CR_SC16P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC16P_SHIFT))&OSC_CR_SC16P_MASK)

#define OSC_CR_SC8P_MASK           0x2u
#define OSC_CR_SC8P_SHIFT          1
#define OSC_CR_SC8P_WIDTH          1
#define OSC_CR_SC8P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC8P_SHIFT))&OSC_CR_SC8P_MASK)

#define OSC_CR_SC4P_MASK           0x4u
#define OSC_CR_SC4P_SHIFT          2
#define OSC_CR_SC4P_WIDTH          1
#define OSC_CR_SC4P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC4P_SHIFT))&OSC_CR_SC4P_MASK)

#define OSC_CR_SC2P_MASK           0x8u
#define OSC_CR_SC2P_SHIFT          3
#define OSC_CR_SC2P_WIDTH          1
#define OSC_CR_SC2P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC2P_SHIFT))&OSC_CR_SC2P_MASK)

#define OSC_CR_EREFSTEN_MASK        0x20u
#define OSC_CR_EREFSTEN_SHIFT       5
#define OSC_CR_EREFSTEN_WIDTH       1

```

```

#define OSC_CR_EREFSTEN(x)
(((uint8_t)((uint8_t)(x)<<OSC_CR_EREFSTEN_SHIFT))&OSC_CR_EREFSTEN_MASK)

#define OSC_CR_ERCLKEN_MASK          0x80u
#define OSC_CR_ERCLKEN_SHIFT         7
#define OSC_CR_ERCLKEN_WIDTH         1

#define OSC_CR_ERCLKEN(x)
(((uint8_t)((uint8_t)(x)<<OSC_CR_ERCLKEN_SHIFT))&OSC_CR_ERCLKEN_MASK)

/*!
 * @}
 */

/* /* end of group OSC_Register_Masks */

```

```

/* OSC - Peripheral instance base addresses */

/** Peripheral OSC0 base address */

#define OSC0_BASE          (0x40065000u)

/** Peripheral OSC0 base pointer */

#define OSC0              ((OSC_Type *)OSCO_BASE)

#define OSC0_BASE_PTR      (OSCO)

/** Array initializer of OSC peripheral base addresses */

#define OSC_BASE_ADDRS     { OSC0_BASE }

/** Array initializer of OSC peripheral base pointers */

#define OSC_BASE_PTRS      { OSC0 }

```

```

/*
-----*
-- OSC - Register accessor macros
----- */

```

```

*/

```

```
* @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros
* @{
*/

```

```
/* OSC - Register instance definitions */
/* OSC0 */
#define OSC0_CR           OSC_CR_REG(OSC0)
```

```
/*!
* @@
*/
/* end of group OSC_Register_Accessor_Macros */
```

```
/*!
* @@
*/
/* end of group OSC_Peripheral_Access_Layer */
```

```
/*
-----  
-- PIT Peripheral Access Layer  
----- */
```

```
/*!
* @addtogroup PIT_Peripheral_Access_Layer PIT Peripheral Access Layer
* @{
*/

```

```
/** PIT - Register Layout Typedef */
```

```

typedef struct {

    __IO uint32_t MCR;           /**< PIT Module Control Register, offset: 0x0 */

    uint8_t RESERVED_0[220];

    __I uint32_t LTMR64H;        /**< PIT Upper Lifetime Timer Register, offset: 0xE0 */

    __I uint32_t LTMR64L;        /**< PIT Lower Lifetime Timer Register, offset: 0xE4 */

    uint8_t RESERVED_1[24];

    struct {                    /* offset: 0x100, array step: 0x10 */

        __IO uint32_t LDVAL;      /**< Timer Load Value Register, array offset: 0x100, array step:
        0x10 */

        __I uint32_t CVAL;       /**< Current Timer Value Register, array offset: 0x104, array step:
        0x10 */

        __IO uint32_t TCTRL;     /**< Timer Control Register, array offset: 0x108, array step: 0x10
    */

        __IO uint32_t TFLG;      /**< Timer Flag Register, array offset: 0x10C, array step: 0x10
    }

} CHANNEL[2];

} PIT_Type, *PIT_MemMapPtr;

```

```

/*
-----  

-- PIT - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros  

* @{
* /

```

```

/* PIT - Register accessors */  

#define PIT_MCR_REG(base)          ((base)->MCR)  

#define PIT_LTMR64H_REG(base)      ((base)->LTMR64H)

```

```
#define PIT_LTMR64L_REG(base) ((base)->LTMR64L)
#define PIT_LDVAL_REG(base,index) ((base)->CHANNEL[index].LDVAL)
#define PIT_LDVAL_COUNT 2
#define PIT_CVAL_REG(base,index) ((base)->CHANNEL[index].CVAL)
#define PIT_CVAL_COUNT 2
#define PIT_TCTRL_REG(base,index) ((base)->CHANNEL[index].TCTRL)
#define PIT_TCTRL_COUNT 2
#define PIT_TFLG_REG(base,index) ((base)->CHANNEL[index].TFLG)
#define PIT_TFLG_COUNT 2
```

```
/*
 * @}
 */
/* /* end of group PIT_Register_Accessor_Macros */
```

```
/*
-----  
-- PIT Register Masks  
----- */
```

```
/*
 * @addtogroup PIT_Register_Masks PIT Register Masks
 * @{
 */

```

```
/* MCR Bit Fields */

#define PIT_MCR_FRZ_MASK 0x1u
#define PIT_MCR_FRZ_SHIFT 0
#define PIT_MCR_FRZ_WIDTH 1
```

```

#define PIT_MCR_FRZ(x)
(((uint32_t)((uint32_t)(x)<<PIT_MCR_FRZ_SHIFT))&PIT_MCR_FRZ_MASK)

#define PIT_MCR_MDIS_MASK          0x2u
#define PIT_MCR_MDIS_SHIFT         1
#define PIT_MCR_MDIS_WIDTH         1

#define PIT_MCR_MDIS(x)
(((uint32_t)((uint32_t)(x)<<PIT_MCR_MDIS_SHIFT))&PIT_MCR_MDIS_MASK)

/* LTMR64H Bit Fields */

#define PIT_LTMR64H_LTH_MASK        0xFFFFFFFFu
#define PIT_LTMR64H_LTH_SHIFT       0
#define PIT_LTMR64H_LTH_WIDTH       32

#define PIT_LTMR64H_LTH(x)
(((uint32_t)((uint32_t)(x)<<PIT_LTMR64H_LTH_SHIFT))&PIT_LTMR64H_LTH_MASK)

/* LTMR64L Bit Fields */

#define PIT_LTMR64L_LTL_MASK        0xFFFFFFFFu
#define PIT_LTMR64L_LTL_SHIFT       0
#define PIT_LTMR64L_LTL_WIDTH       32

#define PIT_LTMR64L_LTL(x)
(((uint32_t)((uint32_t)(x)<<PIT_LTMR64L_LTL_SHIFT))&PIT_LTMR64L_LTL_MASK)

/* LDVAL Bit Fields */

#define PIT_LDVAL_TSV_MASK          0xFFFFFFFFu
#define PIT_LDVAL_TSV_SHIFT         0
#define PIT_LDVAL_TSV_WIDTH         32

#define PIT_LDVAL_TSV(x)
(((uint32_t)((uint32_t)(x)<<PIT_LDVAL_TSV_SHIFT))&PIT_LDVAL_TSV_MASK)

/* CVAL Bit Fields */

#define PIT_CVAL_TVL_MASK          0xFFFFFFFFu
#define PIT_CVAL_TVL_SHIFT         0
#define PIT_CVAL_TVL_WIDTH         32

#define PIT_CVAL_TVL(x)
(((uint32_t)((uint32_t)(x)<<PIT_CVAL_TVL_SHIFT))&PIT_CVAL_TVL_MASK)

```

```

/* TCTRL Bit Fields */

#define PIT_TCTRL_TEN_MASK          0x1u
#define PIT_TCTRL_TEN_SHIFT         0
#define PIT_TCTRL_TEN_WIDTH         1
#define PIT_TCTRL_TEN(x)            (((uint32_t)((uint32_t)(x))<<PIT_TCTRL_TEN_SHIFT))&PIT_TCTRL_TEN_MASK

#define PIT_TCTRL_TIE_MASK          0x2u
#define PIT_TCTRL_TIE_SHIFT         1
#define PIT_TCTRL_TIE_WIDTH         1
#define PIT_TCTRL_TIE(x)            (((uint32_t)((uint32_t)(x))<<PIT_TCTRL_TIE_SHIFT))&PIT_TCTRL_TIE_MASK

#define PIT_TCTRL_CHN_MASK          0x4u
#define PIT_TCTRL_CHN_SHIFT         2
#define PIT_TCTRL_CHN_WIDTH         1
#define PIT_TCTRL_CHN(x)            (((uint32_t)((uint32_t)(x))<<PIT_TCTRL_CHN_SHIFT))&PIT_TCTRL_CHN_MASK

/* TFLG Bit Fields */

#define PIT_TFLG_TIF_MASK           0x1u
#define PIT_TFLG_TIF_SHIFT          0
#define PIT_TFLG_TIF_WIDTH          1
#define PIT_TFLG_TIF(x)             (((uint32_t)((uint32_t)(x))<<PIT_TFLG_TIF_SHIFT))&PIT_TFLG_TIF_MASK

/*!
 * @}
 */

/* /* end of group PIT_Register_Masks */

```

```

/* PIT - Peripheral instance base addresses */
/** Peripheral PIT base address */

```

```

#define PIT_BASE (0x40037000u)
/** Peripheral PIT base pointer */

#define PIT ((PIT_Type *)PIT_BASE)

#define PIT_BASE_PTR (PIT)

/** Array initializer of PIT peripheral base addresses */

#define PIT_BASE_ADDRS { PIT_BASE }

/** Array initializer of PIT peripheral base pointers */

#define PIT_BASE_PTRS { PIT }

/*
-- PIT - Register accessor macros
----- */
/*!
* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros
* @{
*/
/* PIT - Register instance definitions */
/* PIT */

#define PIT_MCR PIT_MCR_REG(PIT)
#define PIT_LTMR64H PIT_LTMR64H_REG(PIT)
#define PIT_LTMR64L PIT_LTMR64L_REG(PIT)
#define PIT_LDVAL0 PIT_LDVAL_REG(PIT,0)
#define PIT_CVAL0 PIT_CVAL_REG(PIT,0)
#define PIT_TCTRL0 PIT_TCTRL_REG(PIT,0)
#define PIT_TFLG0 PIT_TFLG_REG(PIT,0)
#define PIT_LDVAL1 PIT_LDVAL_REG(PIT,1)

```

```

#define PIT_CVAL1          PIT_CVAL_REG(PIT,1)
#define PIT_TCTRL1          PIT_TCTRL_REG(PIT,1)
#define PIT_TFLG1           PIT_TFLG_REG(PIT,1)

/* PIT - Register array accessors */

#define PIT_LDVAL(index)    PIT_LDVAL_REG(PIT,index)
#define PIT_CVAL(index)      PIT_CVAL_REG(PIT,index)
#define PIT_TCTRL(index)     PIT_TCTRL_REG(PIT,index)
#define PIT_TFLG(index)      PIT_TFLG_REG(PIT,index)

/*!
 * @}
 */

/* end of group PIT_Register_Accessor_Macros */

/*!
 * @}
 */

/* end of group PIT_Peripheral_Access_Layer */

/*
-----  

-- PMC Peripheral Access Layer  

----- */
/*!  

 * @addtogroup PMC_Peripheral_Access_Layer PMC Peripheral Access Layer  

 * @{
 * /
```

```
/** PMC - Register Layout Typedef */

typedef struct {

    __IO uint8_t LVDSC1;           /**< Low Voltage Detect Status And Control 1 register, offset: 0x0
*/
    __IO uint8_t LVDSC2;           /**< Low Voltage Detect Status And Control 2 register, offset: 0x1
*/
    __IO uint8_t REGSC;            /**< Regulator Status And Control register, offset: 0x2 */

} PMC_Type, *PMC_MemMapPtr;
```

```
/*
-----  
-- PMC - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros  
* @{  
*/
```

```
/* PMC - Register accessors */  
  
#define PMC_LVDSC1_REG(base)      ((base)->LVDSC1)  
#define PMC_LVDSC2_REG(base)      ((base)->LVDSC2)  
#define PMC_REGSC_REG(base)       ((base)->REGSC)
```

```
/*!  
* @}  
*/ /* end of group PMC_Register_Accessor_Macros */
```

```

/*
-----  

-- PMC Register Masks  

----- */
  
  

/*!  

 * @addtogroup PMC_Register_Masks PMC Register Masks
 * @{
 * /  
  

/* LVDSC1 Bit Fields */  

#define PMC_LVDSC1_LVDV_MASK          0x3u
#define PMC_LVDSC1_LVDV_SHIFT         0
#define PMC_LVDSC1_LVDV_WIDTH         2
#define PMC_LVDSC1_LVDV(x)            (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDV_SHIFT))&PMC_LVDSC1_LVDV_MASK)  

#define PMC_LVDSC1_LVDRE_MASK         0x10u
#define PMC_LVDSC1_LVDRE_SHIFT        4
#define PMC_LVDSC1_LVDRE_WIDTH        1
#define PMC_LVDSC1_LVDRE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDRE_SHIFT))&PMC_LVDSC1_LVDRE_MASK)  

#define PMC_LVDSC1_LVDIE_MASK         0x20u
#define PMC_LVDSC1_LVDIE_SHIFT        5
#define PMC_LVDSC1_LVDIE_WIDTH        1
#define PMC_LVDSC1_LVDIE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDIE_SHIFT))&PMC_LVDSC1_LVDIE_MASK)  

#define PMC_LVDSC1_LVDACK_MASK        0x40u
#define PMC_LVDSC1_LVDACK_SHIFT       6
#define PMC_LVDSC1_LVDACK_WIDTH       1
#define PMC_LVDSC1_LVDACK(x)          (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDACK_SHIFT))&PMC_LVDSC1_LVDACK_MASK)

```

```

#define PMC_LVDSC1_LVDF_MASK          0x80u
#define PMC_LVDSC1_LVDF_SHIFT         7
#define PMC_LVDSC1_LVDF_WIDTH         1
#define PMC_LVDSC1_LVDF(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDF_SHIFT))&PMC_LVDSC1_LVDF_MASK)

/* LVDSC2 Bit Fields */

#define PMC_LVDSC2_LVWV_MASK          0x3u
#define PMC_LVDSC2_LVWV_SHIFT         0
#define PMC_LVDSC2_LVWV_WIDTH         2
#define PMC_LVDSC2_LVWV(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWV_SHIFT))&PMC_LVDSC2_LVWV_MASK)

#define PMC_LVDSC2_LVWIE_MASK         0x20u
#define PMC_LVDSC2_LVWIE_SHIFT        5
#define PMC_LVDSC2_LVWIE_WIDTH        1
#define PMC_LVDSC2_LVWIE(x)          (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWIE_SHIFT))&PMC_LVDSC2_LVWIE_MASK)

#define PMC_LVDSC2_LVWACK_MASK        0x40u
#define PMC_LVDSC2_LVWACK_SHIFT       6
#define PMC_LVDSC2_LVWACK_WIDTH       1
#define PMC_LVDSC2_LVWACK(x)          (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWACK_SHIFT))&PMC_LVDSC2_LVWACK_MASK)

#define PMC_LVDSC2_LVWF_MASK          0x80u
#define PMC_LVDSC2_LVWF_SHIFT         7
#define PMC_LVDSC2_LVWF_WIDTH         1
#define PMC_LVDSC2_LVWF(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWF_SHIFT))&PMC_LVDSC2_LVWF_MASK)

/* REGSC Bit Fields */

#define PMC_REGSC_BGBE_MASK          0x1u
#define PMC_REGSC_BGBE_SHIFT          0
#define PMC_REGSC_BGBE_WIDTH          1

```

```

#define PMC_REGSC_BGBE(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGBE_SHIFT))&PMC_REGSC_BGBE_MASK)

#define PMC_REGSC_REGONS_MASK          0x4u
#define PMC_REGSC_REGONS_SHIFT        2
#define PMC_REGSC_REGONS_WIDTH        1

#define PMC_REGSC_REGONS(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_REGONS_SHIFT))&PMC_REGSC_REGONS_MASK)

#define PMC_REGSC_ACKISO_MASK         0x8u
#define PMC_REGSC_ACKISO_SHIFT        3
#define PMC_REGSC_ACKISO_WIDTH        1

#define PMC_REGSC_ACKISO(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_ACKISO_SHIFT))&PMC_REGSC_ACKISO_MASK)

#define PMC_REGSC_BGEN_MASK           0x10u
#define PMC_REGSC_BGEN_SHIFT          4
#define PMC_REGSC_BGEN_WIDTH          1

#define PMC_REGSC_BGEN(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGEN_SHIFT))&PMC_REGSC_BGEN_MASK)

/*
 * @}
 */

/* /* end of group PMC_Register_Masks */

```

```

/* PMC - Peripheral instance base addresses */

/** Peripheral PMC base address */

#define PMC_BASE                  (0x4007D000u)

/** Peripheral PMC base pointer */

#define PMC                      ((PMC_Type *)PMC_BASE)

#define PMC_BASE_PTR              (PMC)

/** Array initializer of PMC peripheral base addresses */

```

```

#define PMC_BASE_ADDRS          { PMC_BASE }

/** Array initializer of PMC peripheral base pointers */

#define PMC_BASE_PTRS           { PMC }

/*
-----  

-- PMC - Register accessor macros  

----- */
/*!  

 * @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros  

 * @{
 * /  

  

/* PMC - Register instance definitions */  

/* PMC */  

#define PMC_LVDSC1              PMC_LVDSC1_REG(PMC)  

#define PMC_LVDSC2              PMC_LVDSC2_REG(PMC)  

#define PMC_REGSC               PMC_REGSC_REG(PMC)  

  

/*!  

 * @}  

 */ /* end of group PMC_Register_Accessor_Macros */  

  

/*!  

 * @}  

 */ /* end of group PMC_Peripheral_Access_Layer */

```

```

/*
-----  

-- PORT Peripheral Access Layer  

----- */
  
  

/*!  

 * @addtogroup PORT_Peripheral_Access_Layer PORT Peripheral Access Layer
 * @{
 * /  
  

/** PORT - Register Layout Typedef */  

typedef struct {  

    __IO uint32_t PCR[32];           /**< Pin Control Register n, array offset: 0x0, array step: 0x4 */  

    __O  uint32_t GPCLR;            /**< Global Pin Control Low Register, offset: 0x80 */  

    __O  uint32_t GPCHR;            /**< Global Pin Control High Register, offset: 0x84 */  

    uint8_t RESERVED_0[24];  

    __IO uint32_t ISFR;             /**< Interrupt Status Flag Register, offset: 0xA0 */  

} PORT_Type, *PORT_MemMapPtr;  
  

/*
-----  

-- PORT - Register accessor macros  

----- */
  
  

/*!  

 * @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros
 * @{
 * /
```

```
/* PORT - Register accessors */

#define PORT_PCR_REG(base,index)      ((base)->PCR[index])
#define PORT_PCR_COUNT                32
#define PORT_GPCLR_REG(base)          ((base)->GPCLR)
#define PORT_GPCHR_REG(base)          ((base)->GPCHR)
#define PORT_ISFR_REG(base)           ((base)->ISFR)
```

```
/*
 * @}
 */
/* /* end of group PORT_Register_Accessor_Macros */
```

```
/*
-----  
-- PORT Register Masks  
----- */
```

```
/*
* @addtogroup PORT_Register_Masks PORT Register Masks
* @{
*/

```

```
/* PCR Bit Fields */

#define PORT_PCR_PS_MASK            0x1u
#define PORT_PCR_PS_SHIFT           0
#define PORT_PCR_PS_WIDTH           1
#define PORT_PCR_PS(x)              (((uint32_t)((uint32_t)(x)<<PORT_PCR_PS_SHIFT))&PORT_PCR_PS_MASK)
#define PORT_PCR_PE_MASK            0x2u
#define PORT_PCR_PE_SHIFT           1
```

```

#define PORT_PCR_PE_WIDTH          1
#define PORT_PCR_PE(x)             (((uint32_t)((uint32_t)(x))<<PORT_PCR_PE_SHIFT))&PORT_PCR_PE_MASK
#define PORT_PCR_SRE_MASK          0x4u
#define PORT_PCR_SRE_SHIFT         2
#define PORT_PCR_SRE_WIDTH         1
#define PORT_PCR_SRE(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_SRE_SHIFT))&PORT_PCR_SRE_MASK
#define PORT_PCR_PFE_MASK          0x10u
#define PORT_PCR_PFE_SHIFT         4
#define PORT_PCR_PFE_WIDTH         1
#define PORT_PCR_PFE(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_PFE_SHIFT))&PORT_PCR_PFE_MASK
#define PORT_PCR_DSE_MASK          0x40u
#define PORT_PCR_DSE_SHIFT         6
#define PORT_PCR_DSE_WIDTH         1
#define PORT_PCR_DSE(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_DSE_SHIFT))&PORT_PCR_DSE_MASK
#define PORT_PCR_MUX_MASK          0x700u
#define PORT_PCR_MUX_SHIFT         8
#define PORT_PCR_MUX_WIDTH         3
#define PORT_PCR_MUX(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_MUX_SHIFT))&PORT_PCR_MUX_MASK
#define PORT_PCR_IRQC_MASK          0xF0000u
#define PORT_PCR_IRQC_SHIFT         16
#define PORT_PCR_IRQC_WIDTH         4
#define PORT_PCR_IRQC(x)           (((uint32_t)((uint32_t)(x))<<PORT_PCR_IRQC_SHIFT))&PORT_PCR_IRQC_MASK
#define PORT_PCR_ISF_MASK           0x1000000u
#define PORT_PCR_ISF_SHIFT          24
#define PORT_PCR_ISF_WIDTH          1

```

```

#define PORT_PCR_ISF(x)
(((uint32_t)((uint32_t)(x)<<PORT_PCR_ISF_SHIFT))&PORT_PCR_ISF_MASK)

/* GPCLR Bit Fields */

#define PORT_GPCLR_GPWD_MASK          0xFFFFFu
#define PORT_GPCLR_GPWD_SHIFT         0
#define PORT_GPCLR_GPWD_WIDTH         16
#define PORT_GPCLR_GPWD(x)
(((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWD_SHIFT))&PORT_GPCLR_GPWD_MASK)

#define PORT_GPCLR_GPWE_MASK          0xFFFF0000u
#define PORT_GPCLR_GPWE_SHIFT         16
#define PORT_GPCLR_GPWE_WIDTH         16
#define PORT_GPCLR_GPWE(x)
(((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWE_SHIFT))&PORT_GPCLR_GPWE_MASK)

/* GPCHR Bit Fields */

#define PORT_GPCHR_GPWD_MASK          0xFFFFFu
#define PORT_GPCHR_GPWD_SHIFT         0
#define PORT_GPCHR_GPWD_WIDTH         16
#define PORT_GPCHR_GPWD(x)
(((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWD_SHIFT))&PORT_GPCHR_GPWD_MASK)

#define PORT_GPCHR_GPWE_MASK          0xFFFF0000u
#define PORT_GPCHR_GPWE_SHIFT         16
#define PORT_GPCHR_GPWE_WIDTH         16
#define PORT_GPCHR_GPWE(x)
(((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWE_SHIFT))&PORT_GPCHR_GPWE_MASK)

/* ISFR Bit Fields */

#define PORT_ISFR_ISF_MASK            0xFFFFFFFFu
#define PORT_ISFR_ISF_SHIFT           0
#define PORT_ISFR_ISF_WIDTH           32
#define PORT_ISFR_ISF(x)
(((uint32_t)((uint32_t)(x)<<PORT_ISFR_ISF_SHIFT))&PORT_ISFR_ISF_MASK)

```

```

/*!
* @}

/* /* end of group PORT_Register_Masks */

/* PORT - Peripheral instance base addresses */

/** Peripheral PORTA base address */

#define PORTA_BASE          (0x40049000u)

/** Peripheral PORTA base pointer */

#define PORTA              ((PORT_Type *)PORTA_BASE)

#define PORTA_BASE_PTR      (PORTA)

/** Peripheral PORTB base address */

#define PORTB_BASE          (0x4004A000u)

/** Peripheral PORTB base pointer */

#define PORTB              ((PORT_Type *)PORTB_BASE)

#define PORTB_BASE_PTR      (PORTB)

/** Peripheral PORTC base address */

#define PORTC_BASE          (0x4004B000u)

/** Peripheral PORTC base pointer */

#define PORTC              ((PORT_Type *)PORTC_BASE)

#define PORTC_BASE_PTR      (PORTC)

/** Peripheral PORTD base address */

#define PORTD_BASE          (0x4004C000u)

/** Peripheral PORTD base pointer */

#define PORTD              ((PORT_Type *)PORTD_BASE)

#define PORTD_BASE_PTR      (PORTD)

/** Peripheral PORTE base address */

#define PORTE_BASE          (0x4004D000u)

/** Peripheral PORTE base pointer */

```

```

#define PORTE ((PORT_Type *)PORTE_BASE)
#define PORTE_BASE_PTR (PORTE)
/** Array initializer of PORT peripheral base addresses */
#define PORT_BASE_ADDRS { PORTA_BASE, PORTB_BASE, PORTC_BASE, PORTD_BASE,
PORTE_BASE }

/** Array initializer of PORT peripheral base pointers */
#define PORT_BASE_PTRS { PORTA, PORTB, PORTC, PORTD, PORTE }

```

```

/*
-----  

-- PORT - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros  

* @{
* /

```

```

/* PORT - Register instance definitions */
/* PORTA */

#define PORTA_PCR0 PORT_PCR_REG(PORTA,0)
#define PORTA_PCR1 PORT_PCR_REG(PORTA,1)
#define PORTA_PCR2 PORT_PCR_REG(PORTA,2)
#define PORTA_PCR3 PORT_PCR_REG(PORTA,3)
#define PORTA_PCR4 PORT_PCR_REG(PORTA,4)
#define PORTA_PCR5 PORT_PCR_REG(PORTA,5)
#define PORTA_PCR6 PORT_PCR_REG(PORTA,6)
#define PORTA_PCR7 PORT_PCR_REG(PORTA,7)
#define PORTA_PCR8 PORT_PCR_REG(PORTA,8)

```

```
#define PORTA_PCR9           PORT_PCR_REG(PORTA,9)
#define PORTA_PCR10          PORT_PCR_REG(PORTA,10)
#define PORTA_PCR11          PORT_PCR_REG(PORTA,11)
#define PORTA_PCR12          PORT_PCR_REG(PORTA,12)
#define PORTA_PCR13          PORT_PCR_REG(PORTA,13)
#define PORTA_PCR14          PORT_PCR_REG(PORTA,14)
#define PORTA_PCR15          PORT_PCR_REG(PORTA,15)
#define PORTA_PCR16          PORT_PCR_REG(PORTA,16)
#define PORTA_PCR17          PORT_PCR_REG(PORTA,17)
#define PORTA_PCR18          PORT_PCR_REG(PORTA,18)
#define PORTA_PCR19          PORT_PCR_REG(PORTA,19)
#define PORTA_PCR20          PORT_PCR_REG(PORTA,20)
#define PORTA_PCR21          PORT_PCR_REG(PORTA,21)
#define PORTA_PCR22          PORT_PCR_REG(PORTA,22)
#define PORTA_PCR23          PORT_PCR_REG(PORTA,23)
#define PORTA_PCR24          PORT_PCR_REG(PORTA,24)
#define PORTA_PCR25          PORT_PCR_REG(PORTA,25)
#define PORTA_PCR26          PORT_PCR_REG(PORTA,26)
#define PORTA_PCR27          PORT_PCR_REG(PORTA,27)
#define PORTA_PCR28          PORT_PCR_REG(PORTA,28)
#define PORTA_PCR29          PORT_PCR_REG(PORTA,29)
#define PORTA_PCR30          PORT_PCR_REG(PORTA,30)
#define PORTA_PCR31          PORT_PCR_REG(PORTA,31)
#define PORTA_GPCLR          PORT_GPCLR_REG(PORTA)
#define PORTA_GPCHR          PORT_GPCHR_REG(PORTA)
#define PORTA_ISFR           PORT_ISFR_REG(PORTA)
/* PORTB */
#define PORTB_PCR0           PORT_PCR_REG(PORTB,0)
#define PORTB_PCR1           PORT_PCR_REG(PORTB,1)
```

#define PORTB_PCR2	PORT_PCR_REG(PORTB,2)
#define PORTB_PCR3	PORT_PCR_REG(PORTB,3)
#define PORTB_PCR4	PORT_PCR_REG(PORTB,4)
#define PORTB_PCR5	PORT_PCR_REG(PORTB,5)
#define PORTB_PCR6	PORT_PCR_REG(PORTB,6)
#define PORTB_PCR7	PORT_PCR_REG(PORTB,7)
#define PORTB_PCR8	PORT_PCR_REG(PORTB,8)
#define PORTB_PCR9	PORT_PCR_REG(PORTB,9)
#define PORTB_PCR10	PORT_PCR_REG(PORTB,10)
#define PORTB_PCR11	PORT_PCR_REG(PORTB,11)
#define PORTB_PCR12	PORT_PCR_REG(PORTB,12)
#define PORTB_PCR13	PORT_PCR_REG(PORTB,13)
#define PORTB_PCR14	PORT_PCR_REG(PORTB,14)
#define PORTB_PCR15	PORT_PCR_REG(PORTB,15)
#define PORTB_PCR16	PORT_PCR_REG(PORTB,16)
#define PORTB_PCR17	PORT_PCR_REG(PORTB,17)
#define PORTB_PCR18	PORT_PCR_REG(PORTB,18)
#define PORTB_PCR19	PORT_PCR_REG(PORTB,19)
#define PORTB_PCR20	PORT_PCR_REG(PORTB,20)
#define PORTB_PCR21	PORT_PCR_REG(PORTB,21)
#define PORTB_PCR22	PORT_PCR_REG(PORTB,22)
#define PORTB_PCR23	PORT_PCR_REG(PORTB,23)
#define PORTB_PCR24	PORT_PCR_REG(PORTB,24)
#define PORTB_PCR25	PORT_PCR_REG(PORTB,25)
#define PORTB_PCR26	PORT_PCR_REG(PORTB,26)
#define PORTB_PCR27	PORT_PCR_REG(PORTB,27)
#define PORTB_PCR28	PORT_PCR_REG(PORTB,28)
#define PORTB_PCR29	PORT_PCR_REG(PORTB,29)
#define PORTB_PCR30	PORT_PCR_REG(PORTB,30)

```
#define PORTB_PCR31          PORT_PCR_REG(PORTB,31)
#define PORTB_GPCLR            PORT_GPCLR_REG(PORTB)
#define PORTB_GPCHR            PORT_GPCHR_REG(PORTB)
#define PORTB_ISFR              PORT_ISFR_REG(PORTB)

/* PORTC */

#define PORTC_PCR0              PORT_PCR_REG(PORTC,0)
#define PORTC_PCR1              PORT_PCR_REG(PORTC,1)
#define PORTC_PCR2              PORT_PCR_REG(PORTC,2)
#define PORTC_PCR3              PORT_PCR_REG(PORTC,3)
#define PORTC_PCR4              PORT_PCR_REG(PORTC,4)
#define PORTC_PCR5              PORT_PCR_REG(PORTC,5)
#define PORTC_PCR6              PORT_PCR_REG(PORTC,6)
#define PORTC_PCR7              PORT_PCR_REG(PORTC,7)
#define PORTC_PCR8              PORT_PCR_REG(PORTC,8)
#define PORTC_PCR9              PORT_PCR_REG(PORTC,9)
#define PORTC_PCR10             PORT_PCR_REG(PORTC,10)
#define PORTC_PCR11             PORT_PCR_REG(PORTC,11)
#define PORTC_PCR12             PORT_PCR_REG(PORTC,12)
#define PORTC_PCR13             PORT_PCR_REG(PORTC,13)
#define PORTC_PCR14             PORT_PCR_REG(PORTC,14)
#define PORTC_PCR15             PORT_PCR_REG(PORTC,15)
#define PORTC_PCR16             PORT_PCR_REG(PORTC,16)
#define PORTC_PCR17             PORT_PCR_REG(PORTC,17)
#define PORTC_PCR18             PORT_PCR_REG(PORTC,18)
#define PORTC_PCR19             PORT_PCR_REG(PORTC,19)
#define PORTC_PCR20             PORT_PCR_REG(PORTC,20)
#define PORTC_PCR21             PORT_PCR_REG(PORTC,21)
#define PORTC_PCR22             PORT_PCR_REG(PORTC,22)
#define PORTC_PCR23             PORT_PCR_REG(PORTC,23)
```

```
#define PORTC_PCR24      PORT_PCR_REG(PORTC,24)
#define PORTC_PCR25      PORT_PCR_REG(PORTC,25)
#define PORTC_PCR26      PORT_PCR_REG(PORTC,26)
#define PORTC_PCR27      PORT_PCR_REG(PORTC,27)
#define PORTC_PCR28      PORT_PCR_REG(PORTC,28)
#define PORTC_PCR29      PORT_PCR_REG(PORTC,29)
#define PORTC_PCR30      PORT_PCR_REG(PORTC,30)
#define PORTC_PCR31      PORT_PCR_REG(PORTC,31)
#define PORTC_GPCLR      PORT_GPCLR_REG(PORTC)
#define PORTC_GPCHR      PORT_GPCHR_REG(PORTC)
#define PORTC_ISFR      PORT_ISFR_REG(PORTC)

/* PORTD */

#define PORTD_PCR0      PORT_PCR_REG(PORTD,0)
#define PORTD_PCR1      PORT_PCR_REG(PORTD,1)
#define PORTD_PCR2      PORT_PCR_REG(PORTD,2)
#define PORTD_PCR3      PORT_PCR_REG(PORTD,3)
#define PORTD_PCR4      PORT_PCR_REG(PORTD,4)
#define PORTD_PCR5      PORT_PCR_REG(PORTD,5)
#define PORTD_PCR6      PORT_PCR_REG(PORTD,6)
#define PORTD_PCR7      PORT_PCR_REG(PORTD,7)
#define PORTD_PCR8      PORT_PCR_REG(PORTD,8)
#define PORTD_PCR9      PORT_PCR_REG(PORTD,9)
#define PORTD_PCR10     PORT_PCR_REG(PORTD,10)
#define PORTD_PCR11     PORT_PCR_REG(PORTD,11)
#define PORTD_PCR12     PORT_PCR_REG(PORTD,12)
#define PORTD_PCR13     PORT_PCR_REG(PORTD,13)
#define PORTD_PCR14     PORT_PCR_REG(PORTD,14)
#define PORTD_PCR15     PORT_PCR_REG(PORTD,15)
#define PORTD_PCR16     PORT_PCR_REG(PORTD,16)
```

```
#define PORTD_PCR17      PORT_PCR_REG(PORTD,17)
#define PORTD_PCR18      PORT_PCR_REG(PORTD,18)
#define PORTD_PCR19      PORT_PCR_REG(PORTD,19)
#define PORTD_PCR20      PORT_PCR_REG(PORTD,20)
#define PORTD_PCR21      PORT_PCR_REG(PORTD,21)
#define PORTD_PCR22      PORT_PCR_REG(PORTD,22)
#define PORTD_PCR23      PORT_PCR_REG(PORTD,23)
#define PORTD_PCR24      PORT_PCR_REG(PORTD,24)
#define PORTD_PCR25      PORT_PCR_REG(PORTD,25)
#define PORTD_PCR26      PORT_PCR_REG(PORTD,26)
#define PORTD_PCR27      PORT_PCR_REG(PORTD,27)
#define PORTD_PCR28      PORT_PCR_REG(PORTD,28)
#define PORTD_PCR29      PORT_PCR_REG(PORTD,29)
#define PORTD_PCR30      PORT_PCR_REG(PORTD,30)
#define PORTD_PCR31      PORT_PCR_REG(PORTD,31)
#define PORTD_GPCLR      PORT_GPCLR_REG(PORTD)
#define PORTD_GPCHR      PORT_GPCHR_REG(PORTD)
#define PORTD_ISFR      PORT_ISFR_REG(PORTD)

/* PORTE */

#define PORTE_PCR0      PORT_PCR_REG(PORTE,0)
#define PORTE_PCR1      PORT_PCR_REG(PORTE,1)
#define PORTE_PCR2      PORT_PCR_REG(PORTE,2)
#define PORTE_PCR3      PORT_PCR_REG(PORTE,3)
#define PORTE_PCR4      PORT_PCR_REG(PORTE,4)
#define PORTE_PCR5      PORT_PCR_REG(PORTE,5)
#define PORTE_PCR6      PORT_PCR_REG(PORTE,6)
#define PORTE_PCR7      PORT_PCR_REG(PORTE,7)
#define PORTE_PCR8      PORT_PCR_REG(PORTE,8)
#define PORTE_PCR9      PORT_PCR_REG(PORTE,9)
```

#define PORTE_PCR10	PORT_PCR_REG(PORTE,10)
#define PORTE_PCR11	PORT_PCR_REG(PORTE,11)
#define PORTE_PCR12	PORT_PCR_REG(PORTE,12)
#define PORTE_PCR13	PORT_PCR_REG(PORTE,13)
#define PORTE_PCR14	PORT_PCR_REG(PORTE,14)
#define PORTE_PCR15	PORT_PCR_REG(PORTE,15)
#define PORTE_PCR16	PORT_PCR_REG(PORTE,16)
#define PORTE_PCR17	PORT_PCR_REG(PORTE,17)
#define PORTE_PCR18	PORT_PCR_REG(PORTE,18)
#define PORTE_PCR19	PORT_PCR_REG(PORTE,19)
#define PORTE_PCR20	PORT_PCR_REG(PORTE,20)
#define PORTE_PCR21	PORT_PCR_REG(PORTE,21)
#define PORTE_PCR22	PORT_PCR_REG(PORTE,22)
#define PORTE_PCR23	PORT_PCR_REG(PORTE,23)
#define PORTE_PCR24	PORT_PCR_REG(PORTE,24)
#define PORTE_PCR25	PORT_PCR_REG(PORTE,25)
#define PORTE_PCR26	PORT_PCR_REG(PORTE,26)
#define PORTE_PCR27	PORT_PCR_REG(PORTE,27)
#define PORTE_PCR28	PORT_PCR_REG(PORTE,28)
#define PORTE_PCR29	PORT_PCR_REG(PORTE,29)
#define PORTE_PCR30	PORT_PCR_REG(PORTE,30)
#define PORTE_PCR31	PORT_PCR_REG(PORTE,31)
#define PORTE_GPCLR	PORT_GPCLR_REG(PORTE)
#define PORTE_GPCHR	PORT_GPCHR_REG(PORTE)
#define PORTE_ISFR	PORT_ISFR_REG(PORTE)

/* PORT - Register array accessors */

#define PORTA_PCR(index)	PORT_PCR_REG(PORTA,index)
#define PORTB_PCR(index)	PORT_PCR_REG(PORTB,index)

```
#define PORTC_PCR(index)          PORT_PCR_REG(PORTC,index)
#define PORTD_PCR(index)          PORT_PCR_REG(PORTD,index)
#define PORTE_PCR(index)          PORT_PCR_REG(PORTE,index)

/*!
 * @}
 */

/* /* end of group PORT_Register_Accessor_Macros */
```

```
/*!
 * @}
 */

/* /* end of group PORT_Peripheral_Access_Layer */
```

```
/*
-----  
-- RCM Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup RCM_Peripheral_Access_Layer RCM Peripheral Access Layer  
 * @{  
 */
```

```
/** RCM - Register Layout Typedef */  
  
typedef struct {  
    __I uint8_t SRS0;           /**< System Reset Status Register 0, offset: 0x0 */  
    __I uint8_t SRS1;           /**< System Reset Status Register 1, offset: 0x1 */  
    uint8_t RESERVED_0[2];  
    __IO uint8_t RPFC;          /**< Reset Pin Filter Control register, offset: 0x4 */
```

```
    __IO uint8_t RPFW;           /**< Reset Pin Filter Width register, offset: 0x5 */  
} RCM_Type, *RCM_MemMapPtr;
```

```
/* -----  
-- RCM - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros  
* @{  
*/
```

```
/* RCM - Register accessors */  
  
#define RCM_SRS0_REG(base)      ((base)->SRS0)  
#define RCM_SRS1_REG(base)      ((base)->SRS1)  
#define RCM_RPFC_REG(base)      ((base)->RPFC)  
#define RCM_RPFW_REG(base)      ((base)->RPFW)
```

```
/*!  
* @}  
* /* end of group RCM_Register_Accessor_Macros */
```

```
/* -----  
-- RCM Register Masks  
----- */
```

```
/*!
```

```

* @addtogroup RCM_Register_Masks RCM Register Masks

* @{
*

/* SRS0 Bit Fields */

#define RCM_SRS0_WAKEUP_MASK          0x1u
#define RCM_SRS0_WAKEUP_SHIFT         0
#define RCM_SRS0_WAKEUP_WIDTH         1
#define RCM_SRS0_WAKEUP(x) (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WAKEUP_SHIFT))&RCM_SRS0_WAKEUP_MASK)

#define RCM_SRS0_LVD_MASK            0x2u
#define RCM_SRS0_LVD_SHIFT           1
#define RCM_SRS0_LVD_WIDTH           1
#define RCM_SRS0_LVD(x) (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LVD_SHIFT))&RCM_SRS0_LVD_MASK)

#define RCM_SRS0_LOC_MASK            0x4u
#define RCM_SRS0_LOC_SHIFT           2
#define RCM_SRS0_LOC_WIDTH           1
#define RCM_SRS0_LOC(x) (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOC_SHIFT))&RCM_SRS0_LOC_MASK)

#define RCM_SRS0_LOL_MASK             0x8u
#define RCM_SRS0_LOL_SHIFT            3
#define RCM_SRS0_LOL_WIDTH            1
#define RCM_SRS0_LOL(x) (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOL_SHIFT))&RCM_SRS0_LOL_MASK)

#define RCM_SRS0_WDOG_MASK           0x20u
#define RCM_SRS0_WDOG_SHIFT          5
#define RCM_SRS0_WDOG_WIDTH          1
#define RCM_SRS0_WDOG(x) (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WDOG_SHIFT))&RCM_SRS0_WDOG_MASK)

```

```

#define RCM_SRS0_PIN_MASK          0x40u
#define RCM_SRS0_PIN_SHIFT         6
#define RCM_SRS0_PIN_WIDTH         1
#define RCM_SRS0_PIN(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS0_PIN_SHIFT))&RCM_SRS0_PIN_MASK

#define RCM_SRS0_POR_MASK          0x80u
#define RCM_SRS0_POR_SHIFT         7
#define RCM_SRS0_POR_WIDTH         1
#define RCM_SRS0_POR(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS0_POR_SHIFT))&RCM_SRS0_POR_MASK

/* SRS1 Bit Fields */

#define RCM_SRS1_LOCKUP_MASK        0x2u
#define RCM_SRS1_LOCKUP_SHIFT       1
#define RCM_SRS1_LOCKUP_WIDTH       1
#define RCM_SRS1_LOCKUP(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS1_LOCKUP_SHIFT))&RCM_SRS1_LOCKUP_MASK

#define RCM_SRS1_SW_MASK            0x4u
#define RCM_SRS1_SW_SHIFT           2
#define RCM_SRS1_SW_WIDTH           1
#define RCM_SRS1_SW(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS1_SW_SHIFT))&RCM_SRS1_SW_MASK

#define RCM_SRS1_MDM_AP_MASK        0x8u
#define RCM_SRS1_MDM_AP_SHIFT       3
#define RCM_SRS1_MDM_AP_WIDTH       1
#define RCM_SRS1_MDM_AP(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS1_MDM_AP_SHIFT))&RCM_SRS1_MDM_AP_MASK

#define RCM_SRS1_SACKERR_MASK       0x20u
#define RCM_SRS1_SACKERR_SHIFT      5
#define RCM_SRS1_SACKERR_WIDTH      1
#define RCM_SRS1_SACKERR(x) (((uint8_t)((uint8_t)(x))<<RCM_SRS1_SACKERR_SHIFT))&RCM_SRS1_SACKERR_MASK

```

```

/* RPFC Bit Fields */

#define RCM_RPFC_RSTFLTSRW_MASK          0x3u
#define RCM_RPFC_RSTFLTSRW_SHIFT         0
#define RCM_RPFC_RSTFLTSRW_WIDTH         2
#define RCM_RPFC_RSTFLTSRW(x)            (((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSRW_SHIFT))&RCM_RPFC_RSTFLTSRW_MASK)

#define RCM_RPFC_RSTFLTSS_MASK           0x4u
#define RCM_RPFC_RSTFLTSS_SHIFT          2
#define RCM_RPFC_RSTFLTSS_WIDTH          1
#define RCM_RPFC_RSTFLTSS(x)             (((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSS_SHIFT))&RCM_RPFC_RSTFLTSS_MASK)

/* RPFW Bit Fields */

#define RCM_RPFW_RSTFLTSEL_MASK          0x1Fu
#define RCM_RPFW_RSTFLTSEL_SHIFT         0
#define RCM_RPFW_RSTFLTSEL_WIDTH         5
#define RCM_RPFW_RSTFLTSEL(x)            (((uint8_t)((uint8_t)(x)<<RCM_RPFW_RSTFLTSEL_SHIFT))&RCM_RPFW_RSTFLTSEL_MASK)

/*
 * @}
 */

/* end of group RCM_Register_Masks */

```

```

/* RCM - Peripheral instance base addresses */

/** Peripheral RCM base address */

#define RCM_BASE                      (0x4007F000u)

/** Peripheral RCM base pointer */

#define RCM                         ((RCM_Type *)RCM_BASE)
#define RCM_BASE_PTR                  (RCM)

/** Array initializer of RCM peripheral base addresses */

```

```

#define RCM_BASE_ADDRS           { RCM_BASE }
/** Array initializer of RCM peripheral base pointers */

#define RCM_BASE_PTRS            { RCM }

/*
-----  

-- RCM - Register accessor macros  

----- */
/*!  

 * @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros
 * @{
 * /  

  

/* RCM - Register instance definitions */
/* RCM */  

#define RCM_SRS0                 RCM_SRS0_REG(RCM)  

#define RCM_SRS1                 RCM_SRS1_REG(RCM)  

#define RCM_RPFC                 RCM_RPFC_REG(RCM)  

#define RCM_RPFW                 RCM_RPFW_REG(RCM)  

  

/*!  

 * @}  

 */ /* end of group RCM_Register_Accessor_Macros */  

  

/*!  

 * @}  

 */ /* end of group RCM_Peripheral_Access_Layer */

```

```

/*
-- ROM Peripheral Access Layer
----- */
/*!
* @addtogroup ROM_Peripheral_Access_Layer ROM Peripheral Access Layer
* @{
*/
/** ROM - Register Layout Typedef */
typedef struct {
    __I uint32_t ENTRY[3];           /**< Entry, array offset: 0x0, array step: 0x4 */
    __I uint32_t TABLEMARK;          /**< End of Table Marker Register, offset: 0xC */
    uint8_t RESERVED_0[4028];
    __I uint32_t SYSACCESS;          /**< System Access Register, offset: 0xFCC */
    __I uint32_t PERIPHID4;          /**< Peripheral ID Register, offset: 0xFD0 */
    __I uint32_t PERIPHID5;          /**< Peripheral ID Register, offset: 0xFD4 */
    __I uint32_t PERIPHID6;          /**< Peripheral ID Register, offset: 0xFD8 */
    __I uint32_t PERIPHID7;          /**< Peripheral ID Register, offset: 0xFDC */
    __I uint32_t PERIPHID0;          /**< Peripheral ID Register, offset: 0xFE0 */
    __I uint32_t PERIPHID1;          /**< Peripheral ID Register, offset: 0xFE4 */
    __I uint32_t PERIPHID2;          /**< Peripheral ID Register, offset: 0xFE8 */
    __I uint32_t PERIPHID3;          /**< Peripheral ID Register, offset: 0xFEC */
    __I uint32_t COMPID[4];          /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
} ROM_Type, *ROM_MemMapPtr;

```

```

/*
-----  

-- ROM - Register accessor macros  

----- */
  

/*!  

* @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros  

* @{
* /  

  

/* ROM - Register accessors */  

#define ROM_ENTRY_REG(base,index)      ((base)->ENTRY[index])  

#define ROM_ENTRY_COUNT            3  

#define ROM_TABLEMARK_REG(base)       ((base)->TABLEMARK)  

#define ROM_SYSACCESS_REG(base)       ((base)->SYSACCESS)  

#define ROM_PERIPHID4_REG(base)       ((base)->PERIPHID4)  

#define ROM_PERIPHID5_REG(base)       ((base)->PERIPHID5)  

#define ROM_PERIPHID6_REG(base)       ((base)->PERIPHID6)  

#define ROM_PERIPHID7_REG(base)       ((base)->PERIPHID7)  

#define ROM_PERIPHID0_REG(base)       ((base)->PERIPHID0)  

#define ROM_PERIPHID1_REG(base)       ((base)->PERIPHID1)  

#define ROM_PERIPHID2_REG(base)       ((base)->PERIPHID2)  

#define ROM_PERIPHID3_REG(base)       ((base)->PERIPHID3)  

#define ROM_COMPID_REG(base,index)    ((base)->COMPID[index])  

#define ROM_COMPID_COUNT            4  

  

/*!  

* @}
* /* end of group ROM_Register_Accessor_Macros */

```

```

/*
-----  

-- ROM Register Masks  

----- */
  

/*!  

 * @addtogroup ROM_Register_Masks ROM Register Masks  

 * @{  

 */  

  

/* ENTRY Bit Fields */  

#define ROM_ENTRY_ENTRY_MASK      0xFFFFFFFFu  

#define ROM_ENTRY_ENTRY_SHIFT     0  

#define ROM_ENTRY_ENTRY_WIDTH     32  

#define ROM_ENTRY_ENTRY(x)        (((uint32_t)((uint32_t)(x))<<ROM_ENTRY_ENTRY_SHIFT))&ROM_ENTRY_ENTRY_MASK)  

  

/* TABLEMARK Bit Fields */  

#define ROM_TABLEMARK_MARK_MASK   0xFFFFFFFFu  

#define ROM_TABLEMARK_MARK_SHIFT  0  

#define ROM_TABLEMARK_MARK_WIDTH  32  

#define ROM_TABLEMARK_MARK(x)    (((uint32_t)((uint32_t)(x))<<ROM_TABLEMARK_MARK_SHIFT))&ROM_TABLEMARK_MARK_MASK)  

  

/* SYSACCESS Bit Fields */  

#define ROM_SYSACCESS_SYSACCESS_MASK 0xFFFFFFFFu  

#define ROM_SYSACCESS_SYSACCESS_SHIFT 0  

#define ROM_SYSACCESS_SYSACCESS_WIDTH 32  

#define ROM_SYSACCESS_SYSACCESS(x) (((uint32_t)((uint32_t)(x))<<ROM_SYSACCESS_SYSACCESS_SHIFT))&ROM_SYSACCESS_SYSACCESS_MASK)

```

```

/* PERIPHID4 Bit Fields */

#define ROM_PERIPHID4_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID4_PERIPHID_SHIFT     0
#define ROM_PERIPHID4_PERIPHID_WIDTH    32
#define ROM_PERIPHID4_PERIPHID(x)        (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID4_PERIPHID_SHIFT))&ROM_PERIPHID4_PERIPHID_MASK)

/* PERIPHID5 Bit Fields */

#define ROM_PERIPHID5_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID5_PERIPHID_SHIFT     0
#define ROM_PERIPHID5_PERIPHID_WIDTH    32
#define ROM_PERIPHID5_PERIPHID(x)        (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID5_PERIPHID_SHIFT))&ROM_PERIPHID5_PERIPHID_MASK)

/* PERIPHID6 Bit Fields */

#define ROM_PERIPHID6_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID6_PERIPHID_SHIFT     0
#define ROM_PERIPHID6_PERIPHID_WIDTH    32
#define ROM_PERIPHID6_PERIPHID(x)        (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID6_PERIPHID_SHIFT))&ROM_PERIPHID6_PERIPHID_MASK)

/* PERIPHID7 Bit Fields */

#define ROM_PERIPHID7_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID7_PERIPHID_SHIFT     0
#define ROM_PERIPHID7_PERIPHID_WIDTH    32
#define ROM_PERIPHID7_PERIPHID(x)        (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID7_PERIPHID_SHIFT))&ROM_PERIPHID7_PERIPHID_MASK)

/* PERIPHID0 Bit Fields */

#define ROM_PERIPHIDO_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHIDO_PERIPHID_SHIFT     0
#define ROM_PERIPHIDO_PERIPHID_WIDTH    32
#define ROM_PERIPHIDO_PERIPHID(x)        (((uint32_t)((uint32_t)(x))<<ROM_PERIPHIDO_PERIPHID_SHIFT))&ROM_PERIPHIDO_PERIPHID_MASK)

```

```

/* PERIPHID1 Bit Fields */

#define ROM_PERIPHID1_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID1_PERIPHID_SHIFT     0
#define ROM_PERIPHID1_PERIPHID_WIDTH    32
#define ROM_PERIPHID1_PERIPHID(x)        (((uint32_t)((uint32_t)(x)<<ROM_PERIPHID1_PERIPHID_SHIFT))&ROM_PERIPHID1_PERIPHID_MASK)

/* PERIPHID2 Bit Fields */

#define ROM_PERIPHID2_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID2_PERIPHID_SHIFT     0
#define ROM_PERIPHID2_PERIPHID_WIDTH    32
#define ROM_PERIPHID2_PERIPHID(x)        (((uint32_t)((uint32_t)(x)<<ROM_PERIPHID2_PERIPHID_SHIFT))&ROM_PERIPHID2_PERIPHID_MASK)

/* PERIPHID3 Bit Fields */

#define ROM_PERIPHID3_PERIPHID_MASK      0xFFFFFFFFu
#define ROM_PERIPHID3_PERIPHID_SHIFT     0
#define ROM_PERIPHID3_PERIPHID_WIDTH    32
#define ROM_PERIPHID3_PERIPHID(x)        (((uint32_t)((uint32_t)(x)<<ROM_PERIPHID3_PERIPHID_SHIFT))&ROM_PERIPHID3_PERIPHID_MASK)

/* COMPID Bit Fields */

#define ROM_COMPID_COMPID_MASK          0xFFFFFFFFu
#define ROM_COMPID_COMPID_SHIFT         0
#define ROM_COMPID_COMPID_WIDTH        32
#define ROM_COMPID_COMPID(x)           (((uint32_t)((uint32_t)(x)<<ROM_COMPID_COMPID_SHIFT))&ROM_COMPID_COMPID_MASK)

/*
 * @}
 */

/* /* end of group ROM_Register_Masks */

```

```
/* ROM - Peripheral instance base addresses */

/** Peripheral ROM base address */

#define ROM_BASE (0xF0002000u)

/** Peripheral ROM base pointer */

#define ROM ((ROM_Type *)ROM_BASE)

#define ROM_BASE_PTR (ROM)

/** Array initializer of ROM peripheral base addresses */

#define ROM_BASE_ADDRS { ROM_BASE }

/** Array initializer of ROM peripheral base pointers */

#define ROM_BASE_PTRS { ROM }
```

```
/*
-----  
-- ROM - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros  
 * @{  
 */
```

```
/* ROM - Register instance definitions */

/* ROM */

#define ROM_ENTRY0 ROM_ENTRY_REG(ROM,0)
#define ROM_ENTRY1 ROM_ENTRY_REG(ROM,1)
#define ROM_ENTRY2 ROM_ENTRY_REG(ROM,2)
#define ROM_TABLEMARK ROM_TABLEMARK_REG(ROM)
#define ROM_SYSACCESS ROM_SYSACCESS_REG(ROM)
#define ROM_PERIPHID4 ROM_PERIPHID4_REG(ROM)
```

```
#define ROM_PERIPHID5           ROM_PERIPHID5_REG(ROM)
#define ROM_PERIPHID6           ROM_PERIPHID6_REG(ROM)
#define ROM_PERIPHID7           ROM_PERIPHID7_REG(ROM)
#define ROM_PERIPHID0           ROM_PERIPHID0_REG(ROM)
#define ROM_PERIPHID1           ROM_PERIPHID1_REG(ROM)
#define ROM_PERIPHID2           ROM_PERIPHID2_REG(ROM)
#define ROM_PERIPHID3           ROM_PERIPHID3_REG(ROM)
#define ROM_COMPID0             ROM_COMPID_REG(ROM,0)
#define ROM_COMPID1             ROM_COMPID_REG(ROM,1)
#define ROM_COMPID2             ROM_COMPID_REG(ROM,2)
#define ROM_COMPID3             ROM_COMPID_REG(ROM,3)
```

```
/* ROM - Register array accessors */

#define ROM_ENTRY(index)          ROM_ENTRY_REG(ROM,index)
#define ROM_COMPID(index)         ROM_COMPID_REG(ROM,index)
```

```
/*
 * @}
 */
/* end of group ROM_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group ROM_Peripheral_Access_Layer */
```

```
/*
-- RTC Peripheral Access Layer
----- */
```

```

/*!

* @addtogroup RTC_Peripheral_Access_Layer RTC Peripheral Access Layer

* @{

*/



/** RTC - Register Layout Typedef */

typedef struct {

    __IO uint32_t TSR;           /**< RTC Time Seconds Register, offset: 0x0 */
    __IO uint32_t TPR;           /**< RTC Time Prescaler Register, offset: 0x4 */
    __IO uint32_t TAR;           /**< RTC Time Alarm Register, offset: 0x8 */
    __IO uint32_t TCR;           /**< RTC Time Compensation Register, offset: 0xC */
    __IO uint32_t CR;            /**< RTC Control Register, offset: 0x10 */
    __IO uint32_t SR;            /**< RTC Status Register, offset: 0x14 */
    __IO uint32_t LR;            /**< RTC Lock Register, offset: 0x18 */
    __IO uint32_t IER;           /**< RTC Interrupt Enable Register, offset: 0x1C */

} RTC_Type, *RTC_MemMapPtr;

```

```

/*
-----  

-- RTC - Register accessor macros  

----- */

```

```

/*!

* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros

* @{

*/




```

```

/* RTC - Register accessors */

```

```

#define RTC_TSR_REG(base)          ((base)->TSR)
#define RTC_TPR_REG(base)          ((base)->TPR)
#define RTC_TAR_REG(base)          ((base)->TAR)
#define RTC_TCR_REG(base)          ((base)->TCR)
#define RTC_CR_REG(base)           ((base)->CR)
#define RTC_SR_REG(base)           ((base)->SR)
#define RTC_LR_REG(base)           ((base)->LR)
#define RTC_IER_REG(base)          ((base)->IER)

/*!
 * @}
 */

/* /* end of group RTC_Register_Accessor_Macros */

```

```

/*
-----*
-- RTC Register Masks
----- */

```

```

/*!
 * @addtogroup RTC_Register_Masks RTC Register Masks
 * @{
 */

```

```

/* TSR Bit Fields */

#define RTC_TSR_TSRS_MASK          0xFFFFFFFFu
#define RTC_TSR_TSRS_SHIFT         0
#define RTC_TSR_TSRS_WIDTH         32
#define RTC_TSR_TSRS(x)            (((uint32_t)((uint32_t)(x))<<RTC_TSR_TSRS_SHIFT))&RTC_TSR_TSRS_MASK

```

```

/* TPR Bit Fields */

#define RTC_TPR_TPR_MASK          0xFFFFu
#define RTC_TPR_TPR_SHIFT         0
#define RTC_TPR_TPR_WIDTH         16
#define RTC_TPR_TPR(x) (((uint32_t)((uint32_t)(x))<<RTC_TPR_TPR_SHIFT))&RTC_TPR_TPR_MASK

/* TAR Bit Fields */

#define RTC_TAR_TAR_MASK          0xFFFFFFFFu
#define RTC_TAR_TAR_SHIFT         0
#define RTC_TAR_TAR_WIDTH         32
#define RTC_TAR_TAR(x) (((uint32_t)((uint32_t)(x))<<RTC_TAR_TAR_SHIFT))&RTC_TAR_TAR_MASK

/* TCR Bit Fields */

#define RTC_TCR_TCR_MASK          0xFFu
#define RTC_TCR_TCR_SHIFT         0
#define RTC_TCR_TCR_WIDTH         8
#define RTC_TCR_TCR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCR_SHIFT))&RTC_TCR_TCR_MASK

#define RTC_TCR_CIR_MASK          0xFF00u
#define RTC_TCR_CIR_SHIFT         8
#define RTC_TCR_CIR_WIDTH         8
#define RTC_TCR_CIR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIR_SHIFT))&RTC_TCR_CIR_MASK

#define RTC_TCR_TCV_MASK          0xFF0000u
#define RTC_TCR_TCV_SHIFT         16
#define RTC_TCR_TCV_WIDTH         8
#define RTC_TCR_TCV(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCV_SHIFT))&RTC_TCR_TCV_MASK

#define RTC_TCR_CIC_MASK          0xFF000000u
#define RTC_TCR_CIC_SHIFT         24

```

```

#define RTC_TCR_CIC_WIDTH          8

#define RTC_TCR_CIC(x)             (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIC_SHIFT))&RTC_TCR_CIC_MASK

/* CR Bit Fields */

#define RTC_CR_SWR_MASK           0x1u
#define RTC_CR_SWR_SHIFT          0
#define RTC_CR_SWR_WIDTH          1

#define RTC_CR_SWR(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SWR_SHIFT))&RTC_CR_SWR_MASK

#define RTC_CR_WPE_MASK            0x2u
#define RTC_CR_WPE_SHIFT           1
#define RTC_CR_WPE_WIDTH           1

#define RTC_CR_WPE(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_WPE_SHIFT))&RTC_CR_WPE_MASK

#define RTC_CR_SUP_MASK             0x4u
#define RTC_CR_SUP_SHIFT            2
#define RTC_CR_SUP_WIDTH            1

#define RTC_CR_SUP(x)               (((uint32_t)((uint32_t)(x))<<RTC_CR_SUP_SHIFT))&RTC_CR_SUP_MASK

#define RTC_CR_UM_MASK              0x8u
#define RTC_CR_UM_SHIFT              3
#define RTC_CR_UM_WIDTH              1

#define RTC_CR_UM(x)                (((uint32_t)((uint32_t)(x))<<RTC_CR_UM_SHIFT))&RTC_CR_UM_MASK

#define RTC_CR_OSCE_MASK            0x100u
#define RTC_CR_OSCE_SHIFT            8
#define RTC_CR_OSCE_WIDTH            1

#define RTC_CR_OSCE(x)               (((uint32_t)((uint32_t)(x))<<RTC_CR_OSCE_SHIFT))&RTC_CR_OSCE_MASK

#define RTC_CR_CLKO_MASK            0x200u
#define RTC_CR_CLKO_SHIFT            9

```

```

#define RTC_CR_CLKO_WIDTH           1
#define RTC_CR_CLKO(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_CLKO_SHIFT))&RTC_CR_CLKO_MASK
#define RTC_CR_SC16P_MASK          0x400u
#define RTC_CR_SC16P_SHIFT         10
#define RTC_CR_SC16P_WIDTH          1
#define RTC_CR_SC16P(x)            (((uint32_t)((uint32_t)(x))<<RTC_CR_SC16P_SHIFT))&RTC_CR_SC16P_MASK
#define RTC_CR_SC8P_MASK           0x800u
#define RTC_CR_SC8P_SHIFT          11
#define RTC_CR_SC8P_WIDTH          1
#define RTC_CR_SC8P(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_SC8P_SHIFT))&RTC_CR_SC8P_MASK
#define RTC_CR_SC4P_MASK           0x1000u
#define RTC_CR_SC4P_SHIFT          12
#define RTC_CR_SC4P_WIDTH          1
#define RTC_CR_SC4P(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_SC4P_SHIFT))&RTC_CR_SC4P_MASK
#define RTC_CR_SC2P_MASK           0x2000u
#define RTC_CR_SC2P_SHIFT          13
#define RTC_CR_SC2P_WIDTH          1
#define RTC_CR_SC2P(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_SC2P_SHIFT))&RTC_CR_SC2P_MASK
/* SR Bit Fields */
#define RTC_SR_TIF_MASK            0x1u
#define RTC_SR_TIF_SHIFT           0
#define RTC_SR_TIF_WIDTH           1
#define RTC_SR_TIF(x)              (((uint32_t)((uint32_t)(x))<<RTC_SR_TIF_SHIFT))&RTC_SR_TIF_MASK
#define RTC_SR_TOF_MASK            0x2u
#define RTC_SR_TOF_SHIFT           1

```

```

#define RTC_SR_TOF_WIDTH          1
#define RTC_SR_TOF(x) (((uint32_t)((uint32_t)(x))<<RTC_SR_TOF_SHIFT))&RTC_SR_TOF_MASK
#define RTC_SR_TAF_MASK           0x4u
#define RTC_SR_TAF_SHIFT          2
#define RTC_SR_TAF_WIDTH          1
#define RTC_SR_TAF(x) (((uint32_t)((uint32_t)(x))<<RTC_SR_TAF_SHIFT))&RTC_SR_TAF_MASK
#define RTC_SR_TCE_MASK            0x10u
#define RTC_SR_TCE_SHIFT           4
#define RTC_SR_TCE_WIDTH          1
#define RTC_SR_TCE(x) (((uint32_t)((uint32_t)(x))<<RTC_SR_TCE_SHIFT))&RTC_SR_TCE_MASK
/* LR Bit Fields */
#define RTC_LR_TCL_MASK           0x8u
#define RTC_LR_TCL_SHIFT           3
#define RTC_LR_TCL_WIDTH          1
#define RTC_LR_TCL(x) (((uint32_t)((uint32_t)(x))<<RTC_LR_TCL_SHIFT))&RTC_LR_TCL_MASK
#define RTC_LR_CRL_MASK            0x10u
#define RTC_LR_CRL_SHIFT           4
#define RTC_LR_CRL_WIDTH          1
#define RTC_LR_CRL(x) (((uint32_t)((uint32_t)(x))<<RTC_LR_CRL_SHIFT))&RTC_LR_CRL_MASK
#define RTC_LR_SRL_MASK            0x20u
#define RTC_LR_SRL_SHIFT           5
#define RTC_LR_SRL_WIDTH          1
#define RTC_LR_SRL(x) (((uint32_t)((uint32_t)(x))<<RTC_LR_SRL_SHIFT))&RTC_LR_SRL_MASK
#define RTC_LR_LRL_MASK            0x40u
#define RTC_LR_LRL_SHIFT           6

```

```

#define RTC_LR_LRL_WIDTH          1
#define RTC_LR_LRL(x)             (((uint32_t)((uint32_t)(x))<<RTC_LR_LRL_SHIFT))&RTC_LR_LRL_MASK)

/* IER Bit Fields */

#define RTC_IER_TIIE_MASK         0x1u
#define RTC_IER_TIIE_SHIFT        0
#define RTC_IER_TIIE_WIDTH        1
#define RTC_IER_TIIE(x)           (((uint32_t)((uint32_t)(x))<<RTC_IER_TIIE_SHIFT))&RTC_IER_TIIE_MASK)

#define RTC_IER_TOIE_MASK         0x2u
#define RTC_IER_TOIE_SHIFT        1
#define RTC_IER_TOIE_WIDTH        1
#define RTC_IER_TOIE(x)           (((uint32_t)((uint32_t)(x))<<RTC_IER_TOIE_SHIFT))&RTC_IER_TOIE_MASK)

#define RTC_IER_TAIE_MASK         0x4u
#define RTC_IER_TAIE_SHIFT        2
#define RTC_IER_TAIE_WIDTH        1
#define RTC_IER_TAIE(x)           (((uint32_t)((uint32_t)(x))<<RTC_IER_TAIE_SHIFT))&RTC_IER_TAIE_MASK)

#define RTC_IER_TSIE_MASK         0x10u
#define RTC_IER_TSIE_SHIFT        4
#define RTC_IER_TSIE_WIDTH        1
#define RTC_IER_TSIE(x)           (((uint32_t)((uint32_t)(x))<<RTC_IER_TSIE_SHIFT))&RTC_IER_TSIE_MASK)

#define RTC_IER_WPON_MASK         0x80u
#define RTC_IER_WPON_SHIFT        7
#define RTC_IER_WPON_WIDTH        1
#define RTC_IER_WPON(x)           (((uint32_t)((uint32_t)(x))<<RTC_IER_WPON_SHIFT))&RTC_IER_WPON_MASK)

/*

```

```

* @}

/* /* end of group RTC_Register_Masks */

/* RTC - Peripheral instance base addresses */

/** Peripheral RTC base address */

#define RTC_BASE          (0x4003D000u)

/** Peripheral RTC base pointer */

#define RTC      ((RTC_Type *)RTC_BASE)

#define RTC_BASE_PTR      (RTC)

/** Array initializer of RTC peripheral base addresses */

#define RTC_BASE_ADDRS    { RTC_BASE }

/** Array initializer of RTC peripheral base pointers */

#define RTC_BASE_PTRS     { RTC }

/*
-- RTC - Register accessor macros
----- */
/*!

* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros
* @{
*/
/* RTC - Register instance definitions */

/* RTC */

#define RTC_TSR          RTC_TSR_REG(RTC)
#define RTC_TPR          RTC_TPR_REG(RTC)

```

```
#define RTC_TAR          RTC_TAR_REG(RTC)
#define RTC_TCR          RTC_TCR_REG(RTC)
#define RTC_CR           RTC_CR_REG(RTC)
#define RTC_SR           RTC_SR_REG(RTC)
#define RTC_LR           RTC_LR_REG(RTC)
#define RTC_IER          RTC_IER_REG(RTC)
```

```
/*!
 * @}
 */
/* /* end of group RTC_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* /* end of group RTC_Peripheral_Access_Layer */
```

```
/*
-----  
-- SIM Peripheral Access Layer  
----- */
```

```
/*!  
* @addtogroup SIM_Peripheral_Access_Layer SIM Peripheral Access Layer  
* @{  
*/
```

```
/** SIM - Register Layout Typedef */  
typedef struct {  
    __IO uint32_t SOPT1;           /**< System Options Register 1, offset: 0x0 */
```

```

__IO uint32_t SOPT1CFG;           /**< SOPT1 Configuration Register, offset: 0x4 */
    uint8_t RESERVED_0[4092];

__IO uint32_t SOPT2;             /**< System Options Register 2, offset: 0x1004 */
    uint8_t RESERVED_1[4];

__IO uint32_t SOPT4;             /**< System Options Register 4, offset: 0x100C */
    uint8_t RESERVED_2[4];

__IO uint32_t SOPT5;             /**< System Options Register 5, offset: 0x1010 */
    uint8_t RESERVED_3[8];

__I uint32_t SDID;              /**< System Device Identification Register, offset: 0x1024 */
    uint8_t RESERVED_4[12];

__IO uint32_t SCGC4;             /**< System Clock Gating Control Register 4, offset: 0x1034 */
__IO uint32_t SCGC5;             /**< System Clock Gating Control Register 5, offset: 0x1038 */
__IO uint32_t SCGC6;             /**< System Clock Gating Control Register 6, offset: 0x103C */
__IO uint32_t SCGC7;             /**< System Clock Gating Control Register 7, offset: 0x1040 */
__IO uint32_t CLKDIV1;           /**< System Clock Divider Register 1, offset: 0x1044 */
    uint8_t RESERVED_5[4];

__IO uint32_t FCFG1;             /**< Flash Configuration Register 1, offset: 0x104C */
__I uint32_t FCFG2;              /**< Flash Configuration Register 2, offset: 0x1050 */
    uint8_t RESERVED_6[4];

__I uint32_t UIDMH;              /**< Unique Identification Register Mid-High, offset: 0x1058 */
__I uint32_t UIDML;              /**< Unique Identification Register Mid Low, offset: 0x105C */
__I uint32_t UIDL;               /**< Unique Identification Register Low, offset: 0x1060 */
    uint8_t RESERVED_7[156];

__IO uint32_t COPC;              /**< COP Control Register, offset: 0x1100 */
__O uint32_t SRVCOP;             /**< Service COP Register, offset: 0x1104 */

} SIM_Type, *SIM_MemMapPtr;

```

/* -----

```

-- SIM - Register accessor macros
----- */
/*!
* @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros
* @{
*/
/* SIM - Register accessors */

#define SIM_SOPT1_REG(base) ((base)->SOPT1)
#define SIM_SOPT1CFG_REG(base) ((base)->SOPT1CFG)
#define SIM_SOPT2_REG(base) ((base)->SOPT2)
#define SIM_SOPT4_REG(base) ((base)->SOPT4)
#define SIM_SOPT5_REG(base) ((base)->SOPT5)
#define SIM_SOPT7_REG(base) ((base)->SOPT7)
#define SIM_SDID_REG(base) ((base)->SDID)
#define SIM_SCGC4_REG(base) ((base)->SCGC4)
#define SIM_SCGC5_REG(base) ((base)->SCGC5)
#define SIM_SCGC6_REG(base) ((base)->SCGC6)
#define SIM_SCGC7_REG(base) ((base)->SCGC7)
#define SIM_CLKDIV1_REG(base) ((base)->CLKDIV1)
#define SIM_FCFG1_REG(base) ((base)->FCFG1)
#define SIM_FCFG2_REG(base) ((base)->FCFG2)
#define SIM_UIDMH_REG(base) ((base)->UIDMH)
#define SIM_UIDML_REG(base) ((base)->UIDML)
#define SIM_UIDL_REG(base) ((base)->UIDL)
#define SIM_COPC_REG(base) ((base)->COPC)
#define SIM_SRVCOP_REG(base) ((base)->SRVCOP)

```

```
/*!  
 * @}  
 */ /* end of group SIM_Register_Accessor_Macros */
```

```
/* -----  
-- SIM Register Masks  
----- */
```

```
/*!  
 * @addtogroup SIM_Register_Masks SIM Register Masks  
 * @{  
 */
```

```
/* SOPT1 Bit Fields */  
  
#define SIM_SOPT1_OSC32KSEL_MASK      0xC0000u  
#define SIM_SOPT1_OSC32KSEL_SHIFT     18  
#define SIM_SOPT1_OSC32KSEL_WIDTH     2  
  
#define SIM_SOPT1_OSC32KSEL(x)  
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1_OSC32KSEL_SHIFT))&SIM_SOPT1_OSC32KSEL_MASK)  
  
#define SIM_SOPT1_USBVSTBY_MASK       0x20000000u  
#define SIM_SOPT1_USBVSTBY_SHIFT     29  
#define SIM_SOPT1_USBVSTBY_WIDTH     1  
  
#define SIM_SOPT1_USBVSTBY(x)  
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBVSTBY_SHIFT))&SIM_SOPT1_USBVSTBY_MASK)  
  
#define SIM_SOPT1_USBSSTBY_MASK       0x40000000u  
#define SIM_SOPT1_USBSSTBY_SHIFT     30  
#define SIM_SOPT1_USBSSTBY_WIDTH     1
```

```

#define SIM_SOPT1_USBSSTBY(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBSSTBY_SHIFT))&SIM_SOPT1_USBSSTBY_MASK)

#define SIM_SOPT1_USBREGEN_MASK          0x80000000u
#define SIM_SOPT1_USBREGEN_SHIFT        31
#define SIM_SOPT1_USBREGEN_WIDTH        1

#define SIM_SOPT1_USBREGEN(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBREGEN_SHIFT))&SIM_SOPT1_USBREGEN_MASK)

/* SOPT1CFG Bit Fields */

#define SIM_SOPT1CFG_URWE_MASK          0x1000000u
#define SIM_SOPT1CFG_URWE_SHIFT        24
#define SIM_SOPT1CFG_URWE_WIDTH        1

#define SIM_SOPT1CFG_URWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_URWE_SHIFT))&SIM_SOPT1CFG_URWE_MASK)

#define SIM_SOPT1CFG_UVSWE_MASK          0x2000000u
#define SIM_SOPT1CFG_UVSWE_SHIFT        25
#define SIM_SOPT1CFG_UVSWE_WIDTH        1

#define SIM_SOPT1CFG_UVSWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_UVSWE_SHIFT))&SIM_SOPT1CFG_UVSWE_MASK)

#define SIM_SOPT1CFG_USSWE_MASK          0x4000000u
#define SIM_SOPT1CFG_USSWE_SHIFT        26
#define SIM_SOPT1CFG_USSWE_WIDTH        1

#define SIM_SOPT1CFG_USSWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_USSWE_SHIFT))&SIM_SOPT1CFG_USSWE_MASK)

/* SOPT2 Bit Fields */

#define SIM_SOPT2_RTCCLKOUTSEL_MASK      0x10u
#define SIM_SOPT2_RTCCLKOUTSEL_SHIFT    4
#define SIM_SOPT2_RTCCLKOUTSEL_WIDTH    1

#define SIM_SOPT2_RTCCLKOUTSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT2_RTCCLKOUTSEL_SHIFT))&SIM_SOPT2_RTCCLKOUTSEL_MASK)

#define SIM_SOPT2_CLKOUTSEL_MASK          0xE0u
#define SIM_SOPT2_CLKOUTSEL_SHIFT        5

```

```

#define SIM_SOPT2_CLKOUTSEL_WIDTH      3
#define SIM_SOPT2_CLKOUTSEL(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_CLKOUTSEL_SHIFT))&SIM_SOPT2_CLKOUTSEL_MASK)
#define SIM_SOPT2_PLLFLLSEL_MASK      0x10000u
#define SIM_SOPT2_PLLFLLSEL_SHIFT    16
#define SIM_SOPT2_PLLFLLSEL_WIDTH     1
#define SIM_SOPT2_PLLFLLSEL(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_PLLFLLSEL_SHIFT))&SIM_SOPT2_PLLFLLSEL_MASK)
#define SIM_SOPT2_USBSRC_MASK        0x40000u
#define SIM_SOPT2_USBSRC_SHIFT       18
#define SIM_SOPT2_USBSRC_WIDTH       1
#define SIM_SOPT2_USBSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_USBSRC_SHIFT))&SIM_SOPT2_USBSRC_MASK)
#define SIM_SOPT2_TPMSRC_MASK        0x3000000u
#define SIM_SOPT2_TPMSRC_SHIFT       24
#define SIM_SOPT2_TPMSRC_WIDTH       2
#define SIM_SOPT2_TPMSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_TPMSRC_SHIFT))&SIM_SOPT2_TPMSRC_MASK)
#define SIM_SOPT2_UART0SRC_MASK      0xC000000u
#define SIM_SOPT2_UART0SRC_SHIFT     26
#define SIM_SOPT2_UART0SRC_WIDTH     2
#define SIM_SOPT2_UART0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_UART0SRC_SHIFT))&SIM_SOPT2_UART0SRC_MASK)
/* SOPT4 Bit Fields */
#define SIM_SOPT4 TPM1CH0SRC_MASK      0x40000u
#define SIM_SOPT4 TPM1CH0SRC_SHIFT    18
#define SIM_SOPT4 TPM1CH0SRC_WIDTH     1
#define SIM_SOPT4 TPM1CH0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM1CH0SRC_SHIFT))&SIM_SOPT4 TPM1CH0SRC_MASK)
#define SIM_SOPT4 TPM2CH0SRC_MASK      0x100000u
#define SIM_SOPT4 TPM2CH0SRC_SHIFT    20

```

```

#define SIM_SOPT4 TPM2CH0SRC_WIDTH      1
#define SIM_SOPT4 TPM2CH0SRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM2CH0SRC_SHIFT))&SIM_SOPT4 TPM2CH0SRC_MASK)

#define SIM_SOPT4 TPM0CLKSEL_MASK        0x1000000u
#define SIM_SOPT4 TPM0CLKSEL_SHIFT      24
#define SIM_SOPT4 TPM0CLKSEL_WIDTH      1
#define SIM_SOPT4 TPM0CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM0CLKSEL_SHIFT))&SIM_SOPT4 TPM0CLKSEL_MASK)

#define SIM_SOPT4 TPM1CLKSEL_MASK        0x2000000u
#define SIM_SOPT4 TPM1CLKSEL_SHIFT      25
#define SIM_SOPT4 TPM1CLKSEL_WIDTH      1
#define SIM_SOPT4 TPM1CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM1CLKSEL_SHIFT))&SIM_SOPT4 TPM1CLKSEL_MASK)

#define SIM_SOPT4 TPM2CLKSEL_MASK        0x4000000u
#define SIM_SOPT4 TPM2CLKSEL_SHIFT      26
#define SIM_SOPT4 TPM2CLKSEL_WIDTH      1
#define SIM_SOPT4 TPM2CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM2CLKSEL_SHIFT))&SIM_SOPT4 TPM2CLKSEL_MASK)

/* SOPT5 Bit Fields */

#define SIM_SOPT5 UART0TXSRC_MASK        0x3u
#define SIM_SOPT5 UART0TXSRC_SHIFT      0
#define SIM_SOPT5 UART0TXSRC_WIDTH      2
#define SIM_SOPT5 UART0TXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5 UART0TXSRC_SHIFT))&SIM_SOPT5 UART0TXSRC_MASK)

#define SIM_SOPT5 UART0RXSRC_MASK        0x4u
#define SIM_SOPT5 UART0RXSRC_SHIFT      2
#define SIM_SOPT5 UART0RXSRC_WIDTH      1
#define SIM_SOPT5 UART0RXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5 UART0RXSRC_SHIFT))&SIM_SOPT5 UART0RXSRC_MASK)

#define SIM_SOPT5 UART1TXSRC_MASK        0x30u
#define SIM_SOPT5 UART1TXSRC_SHIFT      4

```

```

#define SIM_SOPT5_UART1TXSRC_WIDTH          2
#define SIM_SOPT5_UART1TXSRC(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART1TXSRC_SHIFT))&SIM_SOPT5_UART1TXSRC_MASK)
#define SIM_SOPT5_UART1RXSRC_MASK           0x40u
#define SIM_SOPT5_UART1RXSRC_SHIFT         6
#define SIM_SOPT5_UART1RXSRC_WIDTH         1
#define SIM_SOPT5_UART1RXSRC(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART1RXSRC_SHIFT))&SIM_SOPT5_UART1RXSRC_MASK)
#define SIM_SOPT5_UART0ODE_MASK            0x10000u
#define SIM_SOPT5_UART0ODE_SHIFT          16
#define SIM_SOPT5_UART0ODE_WIDTH          1
#define SIM_SOPT5_UART0ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART0ODE_SHIFT))&SIM_SOPT5_UART0ODE_MASK)
#define SIM_SOPT5_UART1ODE_MASK            0x20000u
#define SIM_SOPT5_UART1ODE_SHIFT          17
#define SIM_SOPT5_UART1ODE_WIDTH          1
#define SIM_SOPT5_UART1ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART1ODE_SHIFT))&SIM_SOPT5_UART1ODE_MASK)
#define SIM_SOPT5_UART2ODE_MASK            0x40000u
#define SIM_SOPT5_UART2ODE_SHIFT          18
#define SIM_SOPT5_UART2ODE_WIDTH          1
#define SIM_SOPT5_UART2ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART2ODE_SHIFT))&SIM_SOPT5_UART2ODE_MASK)
/* SOPT7 Bit Fields */
#define SIM_SOPT7_ADC0TRGSEL_MASK          0xFu
#define SIM_SOPT7_ADC0TRGSEL_SHIFT        0
#define SIM_SOPT7_ADC0TRGSEL_WIDTH        4
#define SIM_SOPT7_ADC0TRGSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADC0TRGSEL_SHIFT))&SIM_SOPT7_ADC0TRGSEL_MASK)
#define SIM_SOPT7_ADCOPRETRGSEL_MASK       0x10u
#define SIM_SOPT7_ADCOPRETRGSEL_SHIFT     4

```

```

#define SIM_SOPT7_ADCOPRETRGSEL_WIDTH      1

#define SIM_SOPT7_ADCOPRETRGSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADCOPRETRGSEL_SHIFT))&SIM_SOPT7_ADCOPRETRGSEL_MASK)

#define SIM_SOPT7_ADCOALTTRGEN_MASK        0x80u
#define SIM_SOPT7_ADCOALTTRGEN_SHIFT      7
#define SIM_SOPT7_ADCOALTTRGEN_WIDTH      1

#define SIM_SOPT7_ADCOALTTRGEN(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADCOALTTRGEN_SHIFT))&SIM_SOPT7_ADCOALTTRGEN_MASK)

/* SDID Bit Fields */

#define SIM_SDID_PINID_MASK              0xFu
#define SIM_SDID_PINID_SHIFT             0
#define SIM_SDID_PINID_WIDTH            4

#define SIM_SDID_PINID(x)
(((uint32_t)((uint32_t)(x)<<SIM_SDID_PINID_SHIFT))&SIM_SDID_PINID_MASK)

#define SIM_SDID_DIEID_MASK             0xF80u
#define SIM_SDID_DIEID_SHIFT            7
#define SIM_SDID_DIEID_WIDTH            5

#define SIM_SDID_DIEID(x)
(((uint32_t)((uint32_t)(x)<<SIM_SDID_DIEID_SHIFT))&SIM_SDID_DIEID_MASK)

#define SIM_SDID_REVID_MASK             0xF000u
#define SIM_SDID_REVID_SHIFT            12
#define SIM_SDID_REVID_WIDTH            4

#define SIM_SDID_REVID(x)
(((uint32_t)((uint32_t)(x)<<SIM_SDID_REVID_SHIFT))&SIM_SDID_REVID_MASK)

#define SIM_SDID_SRAMSIZE_MASK          0xF0000u
#define SIM_SDID_SRAMSIZE_SHIFT         16
#define SIM_SDID_SRAMSIZE_WIDTH         4

#define SIM_SDID_SRAMSIZE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SDID_SRAMSIZE_SHIFT))&SIM_SDID_SRAMSIZE_MASK)

#define SIM_SDID_SERIESID_MASK          0xF00000u

```

```

#define SIM_SDID_SERIESID_SHIFT      20
#define SIM_SDID_SERIESID_WIDTH      4
#define SIM_SDID_SERIESID(x)         (((uint32_t)((uint32_t)(x)<<SIM_SDID_SERIESID_SHIFT))&SIM_SDID_SERIESID_MASK)
#define SIM_SDID_SUBFAMID_MASK       0xF000000u
#define SIM_SDID_SUBFAMID_SHIFT      24
#define SIM_SDID_SUBFAMID_WIDTH      4
#define SIM_SDID_SUBFAMID(x)         (((uint32_t)((uint32_t)(x)<<SIM_SDID_SUBFAMID_SHIFT))&SIM_SDID_SUBFAMID_MASK)
#define SIM_SDID_FAMID_MASK          0xF0000000u
#define SIM_SDID_FAMID_SHIFT         28
#define SIM_SDID_FAMID_WIDTH         4
#define SIM_SDID_FAMID(x)            (((uint32_t)((uint32_t)(x)<<SIM_SDID_FAMID_SHIFT))&SIM_SDID_FAMID_MASK)

/* SCGC4 Bit Fields */

#define SIM_SCGC4_I2C0_MASK          0x40u
#define SIM_SCGC4_I2C0_SHIFT          6
#define SIM_SCGC4_I2C0_WIDTH          1
#define SIM_SCGC4_I2C0(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_I2C0_SHIFT))&SIM_SCGC4_I2C0_MASK)
#define SIM_SCGC4_I2C1_MASK          0x80u
#define SIM_SCGC4_I2C1_SHIFT          7
#define SIM_SCGC4_I2C1_WIDTH          1
#define SIM_SCGC4_I2C1(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_I2C1_SHIFT))&SIM_SCGC4_I2C1_MASK)
#define SIM_SCGC4_UART0_MASK          0x400u
#define SIM_SCGC4_UART0_SHIFT          10
#define SIM_SCGC4_UART0_WIDTH          1
#define SIM_SCGC4_UART0(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_UART0_SHIFT))&SIM_SCGC4_UART0_MASK)
#define SIM_SCGC4_UART1_MASK          0x800u

```

```

#define SIM_SCGC4_UART1_SHIFT          11
#define SIM_SCGC4_UART1_WIDTH          1
#define SIM_SCGC4_UART1(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_UART1_SHIFT))&SIM_SCGC4_UART1_MASK)
#define SIM_SCGC4_UART2_MASK           0x1000u
#define SIM_SCGC4_UART2_SHIFT          12
#define SIM_SCGC4_UART2_WIDTH          1
#define SIM_SCGC4_UART2(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_UART2_SHIFT))&SIM_SCGC4_UART2_MASK)
#define SIM_SCGC4_USBOTG_MASK          0x40000u
#define SIM_SCGC4_USBOTG_SHIFT         18
#define SIM_SCGC4_USBOTG_WIDTH         1
#define SIM_SCGC4_USBOTG(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_USBOTG_SHIFT))&SIM_SCGC4_USBOTG_MASK)
#define SIM_SCGC4_CMP_MASK             0x80000u
#define SIM_SCGC4_CMP_SHIFT            19
#define SIM_SCGC4_CMP_WIDTH            1
#define SIM_SCGC4_CMP(x)               (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_CMP_SHIFT))&SIM_SCGC4_CMP_MASK)
#define SIM_SCGC4_SPI0_MASK            0x400000u
#define SIM_SCGC4_SPI0_SHIFT           22
#define SIM_SCGC4_SPI0_WIDTH          1
#define SIM_SCGC4_SPI0(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI0_SHIFT))&SIM_SCGC4_SPI0_MASK)
#define SIM_SCGC4_SPI1_MASK            0x800000u
#define SIM_SCGC4_SPI1_SHIFT           23
#define SIM_SCGC4_SPI1_WIDTH          1
#define SIM_SCGC4_SPI1(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI1_SHIFT))&SIM_SCGC4_SPI1_MASK)
/* SCGC5 Bit Fields */
#define SIM_SCGC5_LPTMR_MASK           0x1u

```

```

#define SIM_SCGC5_LPTMR_SHIFT          0
#define SIM_SCGC5_LPTMR_WIDTH          1
#define SIM_SCGC5_LPTMR(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_LPTMR_SHIFT))&SIM_SCGC5_LPTMR_MASK)

#define SIM_SCGC5_TSI_MASK            0x20u
#define SIM_SCGC5_TSI_SHIFT           5
#define SIM_SCGC5_TSI_WIDTH           1
#define SIM_SCGC5_TSI(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_TSI_SHIFT))&SIM_SCGC5_TSI_MASK)

#define SIM_SCGC5_PORTA_MASK          0x200u
#define SIM_SCGC5_PORTA_SHIFT         9
#define SIM_SCGC5_PORTA_WIDTH         1
#define SIM_SCGC5_PORTA(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTA_SHIFT))&SIM_SCGC5_PORTA_MASK)

#define SIM_SCGC5_PORTB_MASK          0x400u
#define SIM_SCGC5_PORTB_SHIFT         10
#define SIM_SCGC5_PORTB_WIDTH          1
#define SIM_SCGC5_PORTB(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTB_SHIFT))&SIM_SCGC5_PORTB_MASK)

#define SIM_SCGC5_PORTC_MASK          0x800u
#define SIM_SCGC5_PORTC_SHIFT         11
#define SIM_SCGC5_PORTC_WIDTH          1
#define SIM_SCGC5_PORTC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTC_SHIFT))&SIM_SCGC5_PORTC_MASK)

#define SIM_SCGC5_PORTD_MASK          0x1000u
#define SIM_SCGC5_PORTD_SHIFT         12
#define SIM_SCGC5_PORTD_WIDTH          1
#define SIM_SCGC5_PORTD(x)
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTD_SHIFT))&SIM_SCGC5_PORTD_MASK)

#define SIM_SCGC5 PORTE_MASK          0x2000u
#define SIM_SCGC5 PORTE_SHIFT         13

```

```

#define SIM_SCGC5_PORTE_WIDTH          1

#define SIM_SCGC5_PORTE(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTE_SHIFT))&SIM_SCGC5_PORTE_MASK)

/* SCGC6 Bit Fields */

#define SIM_SCGC6_FTF_MASK           0x1u
#define SIM_SCGC6_FTF_SHIFT          0
#define SIM_SCGC6_FTF_WIDTH          1

#define SIM_SCGC6_FTF(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC6_FTF_SHIFT))&SIM_SCGC6_FTF_MASK)

#define SIM_SCGC6_DMAMUX_MASK        0x2u
#define SIM_SCGC6_DMAMUX_SHIFT       1
#define SIM_SCGC6_DMAMUX_WIDTH       1

#define SIM_SCGC6_DMAMUX(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC6_DMAMUX_SHIFT))&SIM_SCGC6_DMAMUX_MASK)

#define SIM_SCGC6_PIT_MASK           0x800000u
#define SIM_SCGC6_PIT_SHIFT          23
#define SIM_SCGC6_PIT_WIDTH          1

#define SIM_SCGC6_PIT(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC6_PIT_SHIFT))&SIM_SCGC6_PIT_MASK)

#define SIM_SCGC6 TPM0_MASK          0x1000000u
#define SIM_SCGC6 TPM0_SHIFT          24
#define SIM_SCGC6 TPM0_WIDTH          1

#define SIM_SCGC6 TPM0(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM0_SHIFT))&SIM_SCGC6 TPM0_MASK)

#define SIM_SCGC6 TPM1_MASK           0x2000000u
#define SIM_SCGC6 TPM1_SHIFT          25
#define SIM_SCGC6 TPM1_WIDTH          1

#define SIM_SCGC6 TPM1(x) \
(((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM1_SHIFT))&SIM_SCGC6 TPM1_MASK)

#define SIM_SCGC6 TPM2_MASK           0x4000000u
#define SIM_SCGC6 TPM2_SHIFT          26

```

```

#define SIM_SCGC6 TPM2_WIDTH          1
#define SIM_SCGC6 TPM2(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM2_SHIFT))&SIM_SCGC6 TPM2_MASK)
#define SIM_SCGC6 ADC0_MASK           0x8000000u
#define SIM_SCGC6 ADC0_SHIFT          27
#define SIM_SCGC6 ADC0_WIDTH          1
#define SIM_SCGC6 ADC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 ADC0_SHIFT))&SIM_SCGC6 ADC0_MASK)
#define SIM_SCGC6 RTC_MASK            0x20000000u
#define SIM_SCGC6 RTC_SHIFT           29
#define SIM_SCGC6 RTC_WIDTH           1
#define SIM_SCGC6 RTC(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 RTC_SHIFT))&SIM_SCGC6 RTC_MASK)
#define SIM_SCGC6 DAC0_MASK           0x80000000u
#define SIM_SCGC6 DAC0_SHIFT          31
#define SIM_SCGC6 DAC0_WIDTH          1
#define SIM_SCGC6 DAC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 DAC0_SHIFT))&SIM_SCGC6 DAC0_MASK)
/* SCGC7 Bit Fields */
#define SIM_SCGC7 DMA_MASK            0x100u
#define SIM_SCGC7 DMA_SHIFT           8
#define SIM_SCGC7 DMA_WIDTH           1
#define SIM_SCGC7 DMA(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC7 DMA_SHIFT))&SIM_SCGC7 DMA_MASK)
/* CLKDIV1 Bit Fields */
#define SIM_CLKDIV1_OUTDIV4_MASK       0x70000u
#define SIM_CLKDIV1_OUTDIV4_SHIFT      16
#define SIM_CLKDIV1_OUTDIV4_WIDTH      3
#define SIM_CLKDIV1_OUTDIV4(x) (((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV4_SHIFT))&SIM_CLKDIV1_OUTDIV4_MASK)
#define SIM_CLKDIV1_OUTDIV1_MASK       0xF0000000u

```

```

#define SIM_CLKDIV1_OUTDIV1_SHIFT      28
#define SIM_CLKDIV1_OUTDIV1_WIDTH      4
#define SIM_CLKDIV1_OUTDIV1(x)
(((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV1_SHIFT))&SIM_CLKDIV1_OUTDIV1_MASK)

/* FCFG1 Bit Fields */

#define SIM_FCFG1_FLASHDIS_MASK      0x1u
#define SIM_FCFG1_FLASHDIS_SHIFT      0
#define SIM_FCFG1_FLASHDIS_WIDTH      1
#define SIM_FCFG1_FLASHDIS(x)
(((uint32_t)((uint32_t)(x)<<SIM_FCFG1_FLASHDIS_SHIFT))&SIM_FCFG1_FLASHDIS_MASK)

#define SIM_FCFG1_FLASHDOZE_MASK      0x2u
#define SIM_FCFG1_FLASHDOZE_SHIFT      1
#define SIM_FCFG1_FLASHDOZE_WIDTH      1
#define SIM_FCFG1_FLASHDOZE(x)
(((uint32_t)((uint32_t)(x)<<SIM_FCFG1_FLASHDOZE_SHIFT))&SIM_FCFG1_FLASHDOZE_MASK)

#define SIM_FCFG1_PFSIZE_MASK        0xF000000u
#define SIM_FCFG1_PFSIZE_SHIFT        24
#define SIM_FCFG1_PFSIZE_WIDTH        4
#define SIM_FCFG1_PFSIZE(x)
(((uint32_t)((uint32_t)(x)<<SIM_FCFG1_PFSIZE_SHIFT))&SIM_FCFG1_PFSIZE_MASK)

/* FCFG2 Bit Fields */

#define SIM_FCFG2_MAXADDR0_MASK        0x7F000000u
#define SIM_FCFG2_MAXADDR0_SHIFT        24
#define SIM_FCFG2_MAXADDR0_WIDTH        7
#define SIM_FCFG2_MAXADDR0(x)
(((uint32_t)((uint32_t)(x)<<SIM_FCFG2_MAXADDR0_SHIFT))&SIM_FCFG2_MAXADDR0_MASK)

/* UIDMH Bit Fields */

#define SIM_UIDMH_UID_MASK           0xFFFFu
#define SIM_UIDMH_UID_SHIFT           0
#define SIM_UIDMH_UID_WIDTH           16

```

```

#define SIM_UIDMH_UID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_UIDMH_UID_SHIFT))&SIM_UIDMH_UID_MASK)

/* UIDML Bit Fields */

#define SIM_UIDML_UID_MASK          0xFFFFFFFFu
#define SIM_UIDML_UID_SHIFT         0
#define SIM_UIDML_UID_WIDTH         32

#define SIM_UIDML_UID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_UIDML_UID_SHIFT))&SIM_UIDML_UID_MASK)

/* UIDL Bit Fields */

#define SIM_UIDL_UID_MASK          0xFFFFFFFFu
#define SIM_UIDL_UID_SHIFT         0
#define SIM_UIDL_UID_WIDTH         32

#define SIM_UIDL_UID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_UIDL_UID_SHIFT))&SIM_UIDL_UID_MASK)

/* COPC Bit Fields */

#define SIM_COPC_COPW_MASK          0x1u
#define SIM_COPC_COPW_SHIFT         0
#define SIM_COPC_COPW_WIDTH         1

#define SIM_COPC_COPW(x)
(((uint32_t) (((uint32_t)(x))<<SIM_COPC_COPW_SHIFT))&SIM_COPC_COPW_MASK)

#define SIM_COPC_COPCLKS_MASK        0x2u
#define SIM_COPC_COPCLKS_SHIFT       1
#define SIM_COPC_COPCLKS_WIDTH       1

#define SIM_COPC_COPCLKS(x)
(((uint32_t) (((uint32_t)(x))<<SIM_COPC_COPCLKS_SHIFT))&SIM_COPC_COPCLKS_MASK)

#define SIM_COPC_COPT_MASK          0xCu
#define SIM_COPC_COPT_SHIFT         2
#define SIM_COPC_COPT_WIDTH         2

#define SIM_COPC_COPT(x)
(((uint32_t) (((uint32_t)(x))<<SIM_COPC_COPT_SHIFT))&SIM_COPC_COPT_MASK)

/* SRVCOP Bit Fields */

```

```
#define SIM_SRVCOP_SRVCOP_MASK          0xFFu
#define SIM_SRVCOP_SRVCOP_SHIFT         0
#define SIM_SRVCOP_SRVCOP_WIDTH        8
#define SIM_SRVCOP_SRVCOP(x)
(((uint32_t)((uint32_t)(x)<<SIM_SRVCOP_SRVCOP_SHIFT))&SIM_SRVCOP_SRVCOP_MASK)
```

```
/*!
* @}
*/ /* end of group SIM_Register_Masks */
```

```
/* SIM - Peripheral instance base addresses */
/** Peripheral SIM base address */
#define SIM_BASE           (0x40047000u)
/** Peripheral SIM base pointer */
#define SIM               ((SIM_Type *)SIM_BASE)
#define SIM_BASE_PTR       (SIM)
/** Array initializer of SIM peripheral base addresses */
#define SIM_BASE_ADDRS    { SIM_BASE }
/** Array initializer of SIM peripheral base pointers */
#define SIM_BASE_PTRS     { SIM }
```

```
/*
-- SIM - Register accessor macros
----- */
```

```
/*
* @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros
* @{
*/
```

*/

```
/* SIM - Register instance definitions */

/* SIM */

#define SIM_SOPT1           SIM_SOPT1_REG(SIM)
#define SIM_SOPT1CFG        SIM_SOPT1CFG_REG(SIM)
#define SIM_SOPT2           SIM_SOPT2_REG(SIM)
#define SIM_SOPT4           SIM_SOPT4_REG(SIM)
#define SIM_SOPT5           SIM_SOPT5_REG(SIM)
#define SIM_SOPT7           SIM_SOPT7_REG(SIM)
#define SIM_SDID            SIM_SDID_REG(SIM)
#define SIM_SCGC4           SIM_SCGC4_REG(SIM)
#define SIM_SCGC5           SIM_SCGC5_REG(SIM)
#define SIM_SCGC6           SIM_SCGC6_REG(SIM)
#define SIM_SCGC7           SIM_SCGC7_REG(SIM)
#define SIM_CLKDIV1         SIM_CLKDIV1_REG(SIM)
#define SIM_FCFG1           SIM_FCFG1_REG(SIM)
#define SIM_FCFG2           SIM_FCFG2_REG(SIM)
#define SIM_UIDMH          SIM_UIDMH_REG(SIM)
#define SIM_UIDML          SIM_UIDML_REG(SIM)
#define SIM_UIDL            SIM_UIDL_REG(SIM)
#define SIM_COPC            SIM_COPC_REG(SIM)
#define SIM_SRVCOP          SIM_SRVCOP_REG(SIM)

/*
 * @}
 */
/* /* end of group SIM_Register_Accessor_Macros */
```

```

/*!
 * @}
 */

/* /* end of group SIM_Peripheral_Access_Layer */

/*
-----*
-- SMC Peripheral Access Layer
----- */
/*!

 * @addtogroup SMC_Peripheral_Access_Layer SMC Peripheral Access Layer
 * @{
 */

/** SMC - Register Layout Typedef */

typedef struct {

    __IO uint8_t PMPROT;           /**< Power Mode Protection register, offset: 0x0 */
    __IO uint8_t PMCTRL;          /**< Power Mode Control register, offset: 0x1 */
    __IO uint8_t STOPCTRL;         /**< Stop Control Register, offset: 0x2 */
    __I  uint8_t PMSTAT;          /**< Power Mode Status register, offset: 0x3 */

} SMC_Type, *SMC_MemMapPtr;

/*
-----*
-- SMC - Register accessor macros
----- */
/*!

 * @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros
 */

```

```
* @{

/*
 * SMC - Register accessors */

#define SMC_PMPROT_REG(base)      ((base)->PMPROT)
#define SMC_PMCTRL_REG(base)      ((base)->PMCTRL)
#define SMC_STOPCTRL_REG(base)    ((base)->STOPCTRL)
#define SMC_PMSTAT_REG(base)      ((base)->PMSTAT)

/*!
 * @}

/* /* end of group SMC_Register_Accessor_Macros */
```

```
/*
-----  
-- SMC Register Masks  
----- */
```

```
/*!  
* @addtogroup SMC_Register_Masks SMC Register Masks  
* @{

/*
```

```
/* PMPROT Bit Fields */

#define SMC_PMPROT_AVLLS_MASK      0x2u
#define SMC_PMPROT_AVLLS_SHIFT     1
#define SMC_PMPROT_AVLLS_WIDTH     1
```

```

#define SMC_PMPROT_AVLLS(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLLS_SHIFT))&SMC_PMPROT_AVLLS_MASK)

#define SMC_PMPROT_ALLS_MASK          0x8u
#define SMC_PMPROT_ALLS_SHIFT        3
#define SMC_PMPROT_ALLS_WIDTH        1

#define SMC_PMPROT_ALLS(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMPROT_ALLS_SHIFT))&SMC_PMPROT_ALLS_MASK)

#define SMC_PMPROT_AVLP_MASK         0x20u
#define SMC_PMPROT_AVLP_SHIFT       5
#define SMC_PMPROT_AVLP_WIDTH       1

#define SMC_PMPROT_AVLP(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLP_SHIFT))&SMC_PMPROT_AVLP_MASK)

/* PMCTRL Bit Fields */

#define SMC_PMCTRL_STOPM_MASK        0x7u
#define SMC_PMCTRL_STOPM_SHIFT      0
#define SMC_PMCTRL_STOPM_WIDTH      3

#define SMC_PMCTRL_STOPM(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPM_SHIFT))&SMC_PMCTRL_STOPM_MASK)

#define SMC_PMCTRL_STOPA_MASK        0x8u
#define SMC_PMCTRL_STOPA_SHIFT      3
#define SMC_PMCTRL_STOPA_WIDTH      1

#define SMC_PMCTRL_STOPA(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPA_SHIFT))&SMC_PMCTRL_STOPA_MASK)

#define SMC_PMCTRL_RUNM_MASK         0x60u
#define SMC_PMCTRL_RUNM_SHIFT       5
#define SMC_PMCTRL_RUNM_WIDTH       2

#define SMC_PMCTRL_RUNM(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_RUNM_SHIFT))&SMC_PMCTRL_RUNM_MASK)

/* STOPCTRL Bit Fields */

#define SMC_STOPCTRL_VLLSM_MASK     0x7u
#define SMC_STOPCTRL_VLLSM_SHIFT    0

```

```

#define SMC_STOPCTRL_VLLSM_WIDTH      3
#define SMC_STOPCTRL_VLLSM(x) (((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_VLLSM_SHIFT))&SMC_STOPCTRL_VLLSM_MASK)
#define SMC_STOPCTRL_PORPO_MASK       0x20u
#define SMC_STOPCTRL_PORPO_SHIFT     5
#define SMC_STOPCTRL_PORPO_WIDTH     1
#define SMC_STOPCTRL_PORPO(x) (((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PORPO_SHIFT))&SMC_STOPCTRL_PORPO_MASK)
#define SMC_STOPCTRL_PSTOPO_MASK      0xC0u
#define SMC_STOPCTRL_PSTOPO_SHIFT    6
#define SMC_STOPCTRL_PSTOPO_WIDTH    2
#define SMC_STOPCTRL_PSTOPO(x) (((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PSTOPO_SHIFT))&SMC_STOPCTRL_PSTOPO_MASK)

/* PMSTAT Bit Fields */

#define SMC_PMSTAT_PMSTAT_MASK        0x7Fu
#define SMC_PMSTAT_PMSTAT_SHIFT      0
#define SMC_PMSTAT_PMSTAT_WIDTH      7
#define SMC_PMSTAT_PMSTAT(x) (((uint8_t)((uint8_t)(x)<<SMC_PMSTAT_PMSTAT_SHIFT))&SMC_PMSTAT_PMSTAT_MASK)

/*
 * @}
 */

/* end of group SMC_Register_Masks */

```

```

/* SMC - Peripheral instance base addresses */

/** Peripheral SMC base address */
#define SMC_BASE          (0x4007E000u)

/** Peripheral SMC base pointer */
#define SMC              ((SMC_Type *)SMC_BASE)

```

```

#define SMC_BASE_PTR          (SMC)
/** Array initializer of SMC peripheral base addresses */

#define SMC_BASE_ADDRS        { SMC_BASE }

/** Array initializer of SMC peripheral base pointers */

#define SMC_BASE_PTRS          { SMC }

/*
-----*
-- SMC - Register accessor macros
----- */
/*!
* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros
* @{
*/
/* SMC - Register instance definitions */
/* SMC */

#define SMC_PMPROT           SMC_PMPROT_REG(SMC)
#define SMC_PMCTRL            SMC_PMCTRL_REG(SMC)
#define SMC_STOPCTRL          SMC_STOPCTRL_REG(SMC)
#define SMC_PMSTAT             SMC_PMSTAT_REG(SMC)

/*
* @{
*/
/* /* end of group SMC_Register_Accessor_Macros */

/*
*/

```

```

* @}

/* /* end of group SMC_Peripheral_Access_Layer */

/*
-----*
-- SPI Peripheral Access Layer
----- */
/*!

* @addtogroup SPI_Peripheral_Access_Layer SPI Peripheral Access Layer
* @{
*/
/*!
** SPI - Register Layout Typedef */

typedef struct {

    __IO uint8_t C1;           /**< SPI control register 1, offset: 0x0 */
    __IO uint8_t C2;           /**< SPI control register 2, offset: 0x1 */
    __IO uint8_t BR;           /**< SPI baud rate register, offset: 0x2 */
    __IO uint8_t S;             /**< SPI status register, offset: 0x3 */

    uint8_t RESERVED_0[1];

    __IO uint8_t D;             /**< SPI data register, offset: 0x5 */
    uint8_t RESERVED_1[1];

    __IO uint8_t M;             /**< SPI match register, offset: 0x7 */
} SPI_Type, *SPI_MemMapPtr;

/*
-----*
-- SPI - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros  
 * @{  
 */
```

```
/* SPI - Register accessors */  
  
#define SPI_C1_REG(base) ((base)->C1)  
  
#define SPI_C2_REG(base) ((base)->C2)  
  
#define SPI_BR_REG(base) ((base)->BR)  
  
#define SPI_S_REG(base) ((base)->S)  
  
#define SPI_D_REG(base) ((base)->D)  
  
#define SPI_M_REG(base) ((base)->M)
```

```
/*!  
 * @}  
 */ /* end of group SPI_Register_Accessor_Macros */
```

```
/* -----  
-- SPI Register Masks  
----- */
```

```
/*!  
 * @addtogroup SPI_Register_Masks SPI Register Masks  
 * @{  
 */
```

```
/* C1 Bit Fields */
```

```

#define SPI_C1_LSBFE_MASK           0x1u
#define SPI_C1_LSBFE_SHIFT          0
#define SPI_C1_LSBFE_WIDTH          1
#define SPI_C1_LSBFE(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_LSBFE_SHIFT))&SPI_C1_LSBFE_MASK)

#define SPI_C1_SSOE_MASK            0x2u
#define SPI_C1_SSOE_SHIFT           1
#define SPI_C1_SSOE_WIDTH           1
#define SPI_C1_SSOE(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_SSOE_SHIFT))&SPI_C1_SSOE_MASK)

#define SPI_C1_CPHA_MASK             0x4u
#define SPI_C1_CPHA_SHIFT            2
#define SPI_C1_CPHA_WIDTH            1
#define SPI_C1_CPHA(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_CPHA_SHIFT))&SPI_C1_CPHA_MASK)

#define SPI_C1_CPOL_MASK             0x8u
#define SPI_C1_CPOL_SHIFT            3
#define SPI_C1_CPOL_WIDTH            1
#define SPI_C1_CPOL(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_CPOL_SHIFT))&SPI_C1_CPOL_MASK)

#define SPI_C1_MSTR_MASK              0x10u
#define SPI_C1_MSTR_SHIFT             4
#define SPI_C1_MSTR_WIDTH             1
#define SPI_C1_MSTR(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_MSTR_SHIFT))&SPI_C1_MSTR_MASK)

#define SPI_C1_SPTIE_MASK             0x20u
#define SPI_C1_SPTIE_SHIFT             5
#define SPI_C1_SPTIE_WIDTH             1
#define SPI_C1_SPTIE(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_SPTIE_SHIFT))&SPI_C1_SPTIE_MASK)

#define SPI_C1_SPE_MASK                0x40u

```

```

#define SPI_C1_SPE_SHIFT          6
#define SPI_C1_SPE_WIDTH          1
#define SPI_C1_SPE(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_SPE_SHIFT))&SPI_C1_SPE_MASK)
#define SPI_C1_SPIE_MASK          0x80u
#define SPI_C1_SPIE_SHIFT          7
#define SPI_C1_SPIE_WIDTH          1
#define SPI_C1_SPIE(x) (((uint8_t)((uint8_t)(x)<<SPI_C1_SPIE_SHIFT))&SPI_C1_SPIE_MASK)

/* C2 Bit Fields */

#define SPI_C2_SPC0_MASK          0x1u
#define SPI_C2_SPC0_SHIFT          0
#define SPI_C2_SPC0_WIDTH          1
#define SPI_C2_SPC0(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_SPC0_SHIFT))&SPI_C2_SPC0_MASK)

#define SPI_C2_SPISWAI_MASK         0x2u
#define SPI_C2_SPISWAI_SHIFT         1
#define SPI_C2_SPISWAI_WIDTH         1
#define SPI_C2_SPISWAI(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_SPISWAI_SHIFT))&SPI_C2_SPISWAI_MASK)

#define SPI_C2_RXDMAE_MASK          0x4u
#define SPI_C2_RXDMAE_SHIFT          2
#define SPI_C2_RXDMAE_WIDTH          1
#define SPI_C2_RXDMAE(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_RXDMAE_SHIFT))&SPI_C2_RXDMAE_MASK)

#define SPI_C2_BIDIROE_MASK          0x8u
#define SPI_C2_BIDIROE_SHIFT          3
#define SPI_C2_BIDIROE_WIDTH          1
#define SPI_C2_BIDIROE(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_BIDIROE_SHIFT))&SPI_C2_BIDIROE_MASK)

#define SPI_C2_MODFEN_MASK          0x10u

```

```

#define SPI_C2_MODFEN_SHIFT          4
#define SPI_C2_MODFEN_WIDTH          1
#define SPI_C2_MODFEN(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_MODFEN_SHIFT))&SPI_C2_MODFEN_MASK)

#define SPI_C2_TXDMAE_MASK           0x20u
#define SPI_C2_TXDMAE_SHIFT          5
#define SPI_C2_TXDMAE_WIDTH          1
#define SPI_C2_TXDMAE(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_TXDMAE_SHIFT))&SPI_C2_TXDMAE_MASK)

#define SPI_C2_SPMIE_MASK            0x80u
#define SPI_C2_SPMIE_SHIFT           7
#define SPI_C2_SPMIE_WIDTH           1
#define SPI_C2_SPMIE(x) (((uint8_t)((uint8_t)(x)<<SPI_C2_SPMIE_SHIFT))&SPI_C2_SPMIE_MASK)

/* BR Bit Fields */

#define SPI_BR_SPR_MASK              0xFu
#define SPI_BR_SPR_SHIFT             0
#define SPI_BR_SPR_WIDTH             4
#define SPI_BR_SPR(x) (((uint8_t)((uint8_t)(x)<<SPI_BR_SPR_SHIFT))&SPI_BR_SPR_MASK)

#define SPI_BR_SPPR_MASK              0x70u
#define SPI_BR_SPPR_SHIFT             4
#define SPI_BR_SPPR_WIDTH             3
#define SPI_BR_SPPR(x) (((uint8_t)((uint8_t)(x)<<SPI_BR_SPPR_SHIFT))&SPI_BR_SPPR_MASK)

/* S Bit Fields */

#define SPI_S_MODF_MASK              0x10u
#define SPI_S_MODF_SHIFT              4
#define SPI_S_MODF_WIDTH              1
#define SPI_S_MODF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_MODF_SHIFT))&SPI_S_MODF_MASK)

```

```

#define SPI_S_SPTEF_MASK           0x20u
#define SPI_S_SPTEF_SHIFT          5
#define SPI_S_SPTEF_WIDTH          1
#define SPI_S_SPTEF(x)             (((uint8_t)((uint8_t)(x)<<SPI_S_SPTEF_SHIFT))&SPI_S_SPTEF_MASK)

#define SPI_S_SPMF_MASK            0x40u
#define SPI_S_SPMF_SHIFT           6
#define SPI_S_SPMF_WIDTH           1
#define SPI_S_SPMF(x)              (((uint8_t)((uint8_t)(x)<<SPI_S_SPMF_SHIFT))&SPI_S_SPMF_MASK)

#define SPI_S_SPRF_MASK            0x80u
#define SPI_S_SPRF_SHIFT           7
#define SPI_S_SPRF_WIDTH           1
#define SPI_S_SPRF(x)              (((uint8_t)((uint8_t)(x)<<SPI_S_SPRF_SHIFT))&SPI_S_SPRF_MASK)

/* D Bit Fields */

#define SPI_D_Bits_MASK            0xFFu
#define SPI_D_Bits_SHIFT            0
#define SPI_D_Bits_WIDTH            8
#define SPI_D_Bits(x)               (((uint8_t)((uint8_t)(x)<<SPI_D_Bits_SHIFT))&SPI_D_Bits_MASK)

/* M Bit Fields */

#define SPI_M_Bits_MASK            0xFFu
#define SPI_M_Bits_SHIFT            0
#define SPI_M_Bits_WIDTH            8
#define SPI_M_Bits(x)               (((uint8_t)((uint8_t)(x)<<SPI_M_Bits_SHIFT))&SPI_M_Bits_MASK)

*/
* @}

/* /* end of group SPI_Register_Masks */

```

```

/* SPI - Peripheral instance base addresses */

/** Peripheral SPIO base address */

#define SPIO_BASE          (0x40076000u)

/** Peripheral SPIO base pointer */

#define SPIO              ((SPI_Type *)SPIO_BASE)

#define SPIO_BASE_PTR      (SPIO)

/** Peripheral SPI1 base address */

#define SPI1_BASE          (0x40077000u)

/** Peripheral SPI1 base pointer */

#define SPI1              ((SPI_Type *)SPI1_BASE)

#define SPI1_BASE_PTR      (SPI1)

/** Array initializer of SPI peripheral base addresses */

#define SPI_BASE_ADDRS     { SPIO_BASE, SPI1_BASE }

/** Array initializer of SPI peripheral base pointers */

#define SPI_BASE_PTRS       { SPIO, SPI1 }

/*
-----  

-- SPI - Register accessor macros  

----- */

```

```

/*!  

 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros  

 * @{  

 */  


```

```

/* SPI - Register instance definitions */  

/* SPIO */

```

```
#define SPI0_C1          SPI_C1_REG(SPI0)
#define SPI0_C2          SPI_C2_REG(SPI0)
#define SPI0_BR          SPI_BR_REG(SPI0)
#define SPI0_S           SPI_S_REG(SPI0)
#define SPI0_D           SPI_D_REG(SPI0)
#define SPI0_M          SPI_M_REG(SPI0)

/* SPI1 */

#define SPI1_C1          SPI_C1_REG(SPI1)
#define SPI1_C2          SPI_C2_REG(SPI1)
#define SPI1_BR          SPI_BR_REG(SPI1)
#define SPI1_S           SPI_S_REG(SPI1)
#define SPI1_D           SPI_D_REG(SPI1)
#define SPI1_M          SPI_M_REG(SPI1)

/*!

 * @}

 */ /* end of group SPI_Register_Accessor_Macros */
```

```
/*!

 * @}

 */ /* end of group SPI_Peripheral_Access_Layer */
```

```
/*
-----  
-- TPM Peripheral Access Layer  
----- */
```

```
/*!
```

```

* @addtogroup TPM_Peripheral_Access_Layer TPM Peripheral Access Layer

* @{
* /


/** TPM - Register Layout Typedef */

typedef struct {

    __IO uint32_t SC;           /**< Status and Control, offset: 0x0 */
    __IO uint32_t CNT;          /**< Counter, offset: 0x4 */
    __IO uint32_t MOD;          /**< Modulo, offset: 0x8 */

    struct {                  /* offset: 0xC, array step: 0x8 */
        __IO uint32_t CnSC;      /**< Channel (n) Status and Control, array offset: 0xC, array step:
        0x8 */
        __IO uint32_t CnV;       /**< Channel (n) Value, array offset: 0x10, array step: 0x8 */
    } CONTROLS[6];

    uint8_t RESERVED_0[20];

    __IO uint32_t STATUS;       /**< Capture and Compare Status, offset: 0x50 */
    uint8_t RESERVED_1[48];

    __IO uint32_t CONF;         /**< Configuration, offset: 0x84 */
} TPM_Type, *TPM_MemMapPtr;

/* -----
-- TPM - Register accessor macros
----- */

/*!
* @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros
* @{
* /

```

```

/* TPM - Register accessors */

#define TPM_SC_REG(base)          ((base)->SC)
#define TPM_CNT_REG(base)         ((base)->CNT)
#define TPM_MOD_REG(base)         ((base)->MOD)
#define TPM_CnSC_REG(base,index)   ((base)->CONTROLS[index].CnSC)
#define TPM_CnSC_COUNT           6
#define TPM_CnV_REG(base,index)   ((base)->CONTROLS[index].CnV)
#define TPM_CnV_COUNT             6
#define TPM_STATUS_REG(base)       ((base)->STATUS)
#define TPM_CONF_REG(base)         ((base)->CONF)

```

```

/*!
* @}
*/
/* /* end of group TPM_Register_Accessor_Macros */

```

```

/*
-----  

-- TPM Register Masks  

----- */

```

```

/*!  

* @addtogroup TPM_Register_Masks TPM Register Masks  

* @{
* /

```

```

/* SC Bit Fields */

#define TPM_SC_PS_MASK            0x7u
#define TPM_SC_PS_SHIFT            0

```

```

#define TPM_SC_PS_WIDTH          3
#define TPM_SC_PS(x)             (((uint32_t)((uint32_t)(x)<<TPM_SC_PS_SHIFT))&TPM_SC_PS_MASK)
#define TPM_SC_CMOD_MASK         0x18u
#define TPM_SC_CMOD_SHIFT        3
#define TPM_SC_CMOD_WIDTH        2
#define TPM_SC_CMOD(x)           (((uint32_t)((uint32_t)(x)<<TPM_SC_CMOD_SHIFT))&TPM_SC_CMOD_MASK)
#define TPM_SC_CPWMS_MASK         0x20u
#define TPM_SC_CPWMS_SHIFT        5
#define TPM_SC_CPWMS_WIDTH        1
#define TPM_SC_CPWMS(x)           (((uint32_t)((uint32_t)(x)<<TPM_SC_CPWMS_SHIFT))&TPM_SC_CPWMS_MASK)
#define TPM_SC_TOIE_MASK          0x40u
#define TPM_SC_TOIE_SHIFT         6
#define TPM_SC_TOIE_WIDTH         1
#define TPM_SC_TOIE(x)            (((uint32_t)((uint32_t)(x)<<TPM_SC_TOIE_SHIFT))&TPM_SC_TOIE_MASK)
#define TPM_SC_TOF_MASK            0x80u
#define TPM_SC_TOF_SHIFT           7
#define TPM_SC_TOF_WIDTH           1
#define TPM_SC_TOF(x)              (((uint32_t)((uint32_t)(x)<<TPM_SC_TOF_SHIFT))&TPM_SC_TOF_MASK)
#define TPM_SC_DMA_MASK            0x100u
#define TPM_SC_DMA_SHIFT           8
#define TPM_SC_DMA_WIDTH           1
#define TPM_SC_DMA(x)              (((uint32_t)((uint32_t)(x)<<TPM_SC_DMA_SHIFT))&TPM_SC_DMA_MASK)
/* CNT Bit Fields */
#define TPM_CNT_COUNT_MASK         0xFFFFu
#define TPM_CNT_COUNT_SHIFT        0

```

```

#define TPM_CNT_COUNT_WIDTH          16
#define TPM_CNT_COUNT(x)            (((uint32_t)((uint32_t)(x)<<TPM_CNT_COUNT_SHIFT))&TPM_CNT_COUNT_MASK)

/* MOD Bit Fields */

#define TPM_MOD_MOD_MASK           0xFFFFu
#define TPM_MOD_MOD_SHIFT          0
#define TPM_MOD_MOD_WIDTH          16
#define TPM_MOD_MOD(x)             (((uint32_t)((uint32_t)(x)<<TPM_MOD_MOD_SHIFT))&TPM_MOD_MOD_MASK)

/* CnSC Bit Fields */

#define TPM_CnSC_DMA_MASK          0x1u
#define TPM_CnSC_DMA_SHIFT         0
#define TPM_CnSC_DMA_WIDTH         1
#define TPM_CnSC_DMA(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_DMA_SHIFT))&TPM_CnSC_DMA_MASK)

#define TPM_CnSC_ELSA_MASK          0x4u
#define TPM_CnSC_ELSA_SHIFT         2
#define TPM_CnSC_ELSA_WIDTH         1
#define TPM_CnSC_ELSA(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_ELSA_SHIFT))&TPM_CnSC_ELSA_MASK)

#define TPM_CnSC_ELSB_MASK          0x8u
#define TPM_CnSC_ELSB_SHIFT         3
#define TPM_CnSC_ELSB_WIDTH         1
#define TPM_CnSC_ELSB(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_ELSB_SHIFT))&TPM_CnSC_ELSB_MASK)

#define TPM_CnSC_MSA_MASK          0x10u
#define TPM_CnSC_MSA_SHIFT         4
#define TPM_CnSC_MSA_WIDTH         1
#define TPM_CnSC_MSA(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_MSA_SHIFT))&TPM_CnSC_MSA_MASK)

#define TPM_CnSC_MSB_MASK          0x20u

```

```

#define TPM_CnSC_MSB_SHIFT      5
#define TPM_CnSC_MSB_WIDTH      1
#define TPM_CnSC_MSB(x)
(((uint32_t)((uint32_t)(x)<<TPM_CnSC_MSB_SHIFT))&TPM_CnSC_MSB_MASK)

#define TPM_CnSC_CHIE_MASK      0x40u
#define TPM_CnSC_CHIE_SHIFT     6
#define TPM_CnSC_CHIE_WIDTH     1
#define TPM_CnSC_CHIE(x)
(((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHIE_SHIFT))&TPM_CnSC_CHIE_MASK)

#define TPM_CnSC_CHF_MASK       0x80u
#define TPM_CnSC_CHF_SHIFT      7
#define TPM_CnSC_CHF_WIDTH      1
#define TPM_CnSC_CHF(x)
(((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHF_SHIFT))&TPM_CnSC_CHF_MASK)

/* CnV Bit Fields */

#define TPM_CnV_VAL_MASK        0xFFFFu
#define TPM_CnV_VAL_SHIFT       0
#define TPM_CnV_VAL_WIDTH       16
#define TPM_CnV_VAL(x)
(((uint32_t)((uint32_t)(x)<<TPM_CnV_VAL_SHIFT))&TPM_CnV_VAL_MASK)

/* STATUS Bit Fields */

#define TPM_STATUS_CH0F_MASK    0x1u
#define TPM_STATUS_CH0F_SHIFT   0
#define TPM_STATUS_CH0F_WIDTH   1
#define TPM_STATUS_CH0F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH0F_SHIFT))&TPM_STATUS_CH0F_MASK)

#define TPM_STATUS_CH1F_MASK    0x2u
#define TPM_STATUS_CH1F_SHIFT   1
#define TPM_STATUS_CH1F_WIDTH   1
#define TPM_STATUS_CH1F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH1F_SHIFT))&TPM_STATUS_CH1F_MASK)

```

```

#define TPM_STATUS_CH2F_MASK          0x4u
#define TPM_STATUS_CH2F_SHIFT         2
#define TPM_STATUS_CH2F_WIDTH         1
#define TPM_STATUS_CH2F(x) (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH2F_SHIFT))&TPM_STATUS_CH2F_MASK)

#define TPM_STATUS_CH3F_MASK          0x8u
#define TPM_STATUS_CH3F_SHIFT         3
#define TPM_STATUS_CH3F_WIDTH         1
#define TPM_STATUS_CH3F(x) (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH3F_SHIFT))&TPM_STATUS_CH3F_MASK)

#define TPM_STATUS_CH4F_MASK          0x10u
#define TPM_STATUS_CH4F_SHIFT         4
#define TPM_STATUS_CH4F_WIDTH         1
#define TPM_STATUS_CH4F(x) (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH4F_SHIFT))&TPM_STATUS_CH4F_MASK)

#define TPM_STATUS_CH5F_MASK          0x20u
#define TPM_STATUS_CH5F_SHIFT         5
#define TPM_STATUS_CH5F_WIDTH         1
#define TPM_STATUS_CH5F(x) (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH5F_SHIFT))&TPM_STATUS_CH5F_MASK)

#define TPM_STATUS_TOF_MASK           0x100u
#define TPM_STATUS_TOF_SHIFT          8
#define TPM_STATUS_TOF_WIDTH          1
#define TPM_STATUS_TOF(x) (((uint32_t)((uint32_t)(x)<<TPM_STATUS_TOF_SHIFT))&TPM_STATUS_TOF_MASK)

/* CONF Bit Fields */

#define TPM_CONF_DOZEEN_MASK          0x20u
#define TPM_CONF_DOZEEN_SHIFT          5
#define TPM_CONF_DOZEEN_WIDTH          1
#define TPM_CONF_DOZEEN(x) (((uint32_t)((uint32_t)(x)<<TPM_CONF_DOZEEN_SHIFT))&TPM_CONF_DOZEEN_MASK)

```

```

#define TPM_CONF_DBGMODE_MASK          0xC0u
#define TPM_CONF_DBGMODE_SHIFT         6
#define TPM_CONF_DBGMODE_WIDTH         2
#define TPM_CONF_DBGMODE(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_DBGMODE_SHIFT))&TPM_CONF_DBGMODE_MASK)

#define TPM_CONF_GTBEEN_MASK           0x200u
#define TPM_CONF_GTBEEN_SHIFT          9
#define TPM_CONF_GTBEEN_WIDTH          1
#define TPM_CONF_GTBEEN(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_GTBEEN_SHIFT))&TPM_CONF_GTBEEN_MASK)

#define TPM_CONF_CSOT_MASK             0x10000u
#define TPM_CONF_CSOT_SHIFT            16
#define TPM_CONF_CSOT_WIDTH            1
#define TPM_CONF_CSOT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOT_SHIFT))&TPM_CONF_CSOT_MASK)

#define TPM_CONF_CSOO_MASK             0x20000u
#define TPM_CONF_CSOO_SHIFT            17
#define TPM_CONF_CSOO_WIDTH            1
#define TPM_CONF_CSOO(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOO_SHIFT))&TPM_CONF_CSOO_MASK)

#define TPM_CONF_CROT_MASK             0x40000u
#define TPM_CONF_CROT_SHIFT            18
#define TPM_CONF_CROT_WIDTH            1
#define TPM_CONF_CROT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CROT_SHIFT))&TPM_CONF_CROT_MASK)

#define TPM_CONF_TRGSEL_MASK           0xF000000u
#define TPM_CONF_TRGSEL_SHIFT          24
#define TPM_CONF_TRGSEL_WIDTH          4
#define TPM_CONF_TRGSEL(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_TRGSEL_SHIFT))&TPM_CONF_TRGSEL_MASK)

```

```

/*!
 * @}
 */

/* /* end of group TPM_Register_Masks */

/* TPM - Peripheral instance base addresses */

/** Peripheral TPM0 base address */

#define TPM0_BASE          (0x40038000u)

/** Peripheral TPM0 base pointer */

#define TPM0           ((TPM_Type *)TPM0_BASE)

#define TPM0_BASE_PTR    (TPM0)

/** Peripheral TPM1 base address */

#define TPM1_BASE          (0x40039000u)

/** Peripheral TPM1 base pointer */

#define TPM1           ((TPM_Type *)TPM1_BASE)

#define TPM1_BASE_PTR    (TPM1)

/** Peripheral TPM2 base address */

#define TPM2_BASE          (0x4003A000u)

/** Peripheral TPM2 base pointer */

#define TPM2           ((TPM_Type *)TPM2_BASE)

#define TPM2_BASE_PTR    (TPM2)

/** Array initializer of TPM peripheral base addresses */

#define TPM_BASE_ADDRS     { TPM0_BASE, TPM1_BASE, TPM2_BASE }

/** Array initializer of TPM peripheral base pointers */

#define TPM_BASE_PTRS      { TPM0, TPM1, TPM2 }

/*
-----*
-- TPM - Register accessor macros
----- */

```

```

/*!

* @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros

* @{
* /


/* TPM - Register instance definitions */

/* TPM0 */

#define TPM0_SC           TPM_SC_REG(TPM0)
#define TPM0_CNT          TPM_CNT_REG(TPM0)
#define TPM0_MOD          TPM_MOD_REG(TPM0)
#define TPM0_C0SC         TPM_CnSC_REG(TPM0,0)
#define TPM0_C0V          TPM_CnV_REG(TPM0,0)
#define TPM0_C1SC         TPM_CnSC_REG(TPM0,1)
#define TPM0_C1V          TPM_CnV_REG(TPM0,1)
#define TPM0_C2SC         TPM_CnSC_REG(TPM0,2)
#define TPM0_C2V          TPM_CnV_REG(TPM0,2)
#define TPM0_C3SC         TPM_CnSC_REG(TPM0,3)
#define TPM0_C3V          TPM_CnV_REG(TPM0,3)
#define TPM0_C4SC         TPM_CnSC_REG(TPM0,4)
#define TPM0_C4V          TPM_CnV_REG(TPM0,4)
#define TPM0_C5SC         TPM_CnSC_REG(TPM0,5)
#define TPM0_C5V          TPM_CnV_REG(TPM0,5)
#define TPM0_STATUS        TPM_STATUS_REG(TPM0)
#define TPM0_CONF          TPM_CONF_REG(TPM0)

/* TPM1 */

#define TPM1_SC           TPM_SC_REG(TPM1)
#define TPM1_CNT          TPM_CNT_REG(TPM1)

```

```
#define TPM1_MOD           TPM_MOD_REG(TPM1)
#define TPM1_COSC          TPM_CnSC_REG(TPM1,0)
#define TPM1_C0V            TPM_CnV_REG(TPM1,0)
#define TPM1_C1SC           TPM_CnSC_REG(TPM1,1)
#define TPM1_C1V            TPM_CnV_REG(TPM1,1)
#define TPM1_STATUS         TPM_STATUS_REG(TPM1)
#define TPM1_CONF            TPM_CONF_REG(TPM1)
```

```
/* TPM2 */
```

```
#define TPM2_SC            TPM_SC_REG(TPM2)
#define TPM2_CNT             TPM_CNT_REG(TPM2)
#define TPM2_MOD             TPM_MOD_REG(TPM2)
#define TPM2_COSC            TPM_CnSC_REG(TPM2,0)
#define TPM2_C0V              TPM_CnV_REG(TPM2,0)
#define TPM2_C1SC           TPM_CnSC_REG(TPM2,1)
#define TPM2_C1V            TPM_CnV_REG(TPM2,1)
#define TPM2_STATUS          TPM_STATUS_REG(TPM2)
#define TPM2_CONF            TPM_CONF_REG(TPM2)
```

```
/* TPM - Register array accessors */
```

```
#define TPM0_CnSC(index)    TPM_CnSC_REG(TPM0,index)
#define TPM1_CnSC(index)    TPM_CnSC_REG(TPM1,index)
#define TPM2_CnSC(index)    TPM_CnSC_REG(TPM2,index)
#define TPM0_CnV(index)     TPM_CnV_REG(TPM0,index)
#define TPM1_CnV(index)     TPM_CnV_REG(TPM1,index)
#define TPM2_CnV(index)     TPM_CnV_REG(TPM2,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group TPM_Register_Accessor_Macros */
```

```

/*!
 * @}
 */

/* /* end of group TPM_Peripheral_Access_Layer */

/*
-----*
-- TSI Peripheral Access Layer
----- */
/*!

 * @addtogroup TSI_Peripheral_Access_Layer TSI Peripheral Access Layer
 * @{
 */

/** TSI - Register Layout Typedef */
typedef struct {
    __IO uint32_t GENCS;           /**< TSI General Control and Status Register, offset: 0x0 */
    __IO uint32_t DATA;            /**< TSI DATA Register, offset: 0x4 */
    __IO uint32_t TSHD;            /**< TSI Threshold Register, offset: 0x8 */
} TSI_Type, *TSI_MemMapPtr;

/*
-----*
-- TSI - Register accessor macros
----- */
/*!

 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros
 */

```

```

* @@
*/
/* TSI - Register accessors */

#define TSI_GENCS_REG(base)      ((base)->GENCS)
#define TSI_DATA_REG(base)       ((base)->DATA)
#define TSI_TS HD_REG(base)     ((base)->TS HD)

/*!
* @@
*/
/* /* end of group TSI_Register_Accessor_Macros */

/*
----- --
-- TSI Register Masks
----- */
/*!

* @addtogroup TSI_Register_Masks TSI Register Masks
* @@
*/
/* GENCS Bit Fields */

#define TSI_GENCS_CURSW_MASK      0x2u
#define TSI_GENCS_CURSW_SHIFT     1
#define TSI_GENCS_CURSW_WIDTH     1

#define TSI_GENCS_CURSW(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_CURSW_SHIFT))&TSI_GENCS_CURSW_MASK)

```

```

#define TSI_GENCS_EOSF_MASK          0x4u
#define TSI_GENCS_EOSF_SHIFT         2
#define TSI_GENCS_EOSF_WIDTH         1
#define TSI_GENCS_EOSF(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_EOSF_SHIFT))&TSI_GENCS_EOSF_MASK)

#define TSI_GENCS_SCNIP_MASK         0x8u
#define TSI_GENCS_SCNIP_SHIFT        3
#define TSI_GENCS_SCNIP_WIDTH        1
#define TSI_GENCS_SCNIP(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_SCNIP_SHIFT))&TSI_GENCS_SCNIP_MASK)

#define TSI_GENCS_STM_MASK           0x10u
#define TSI_GENCS_STM_SHIFT          4
#define TSI_GENCS_STM_WIDTH          1
#define TSI_GENCS_STM(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_STM_SHIFT))&TSI_GENCS_STM_MASK)

#define TSI_GENCS_STPE_MASK          0x20u
#define TSI_GENCS_STPE_SHIFT         5
#define TSI_GENCS_STPE_WIDTH         1
#define TSI_GENCS_STPE(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_STPE_SHIFT))&TSI_GENCS_STPE_MASK)

#define TSI_GENCS_TSIIEN_MASK         0x40u
#define TSI_GENCS_TSIIEN_SHIFT        6
#define TSI_GENCS_TSIIEN_WIDTH        1
#define TSI_GENCS_TSIIEN(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIIEN_SHIFT))&TSI_GENCS_TSIIEN_MASK)

#define TSI_GENCS_TSIEN_MASK          0x80u
#define TSI_GENCS_TSIEN_SHIFT         7
#define TSI_GENCS_TSIEN_WIDTH         1
#define TSI_GENCS_TSIEN(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIEN_SHIFT))&TSI_GENCS_TSIEN_MASK)

#define TSI_GENCS_NSCN_MASK           0x1F00u

```

```

#define TSI_GENCS_NSCN_SHIFT          8
#define TSI_GENCS_NSCN_WIDTH          5
#define TSI_GENCS_NSCN(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_NSCN_SHIFT))&TSI_GENCS_NSCN_MASK)

#define TSI_GENCS_PS_MASK            0xE000u
#define TSI_GENCS_PS_SHIFT           13
#define TSI_GENCS_PS_WIDTH           3
#define TSI_GENCS_PS(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_PS_SHIFT))&TSI_GENCS_PS_MASK)

#define TSI_GENCS_EXTCHRG_MASK        0x70000u
#define TSI_GENCS_EXTCHRG_SHIFT       16
#define TSI_GENCS_EXTCHRG_WIDTH       3
#define TSI_GENCS_EXTCHRG(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_EXTCHRG_SHIFT))&TSI_GENCS_EXTCHRG_MASK)

#define TSI_GENCS_DVOLT_MASK          0x180000u
#define TSI_GENCS_DVOLT_SHIFT         19
#define TSI_GENCS_DVOLT_WIDTH         2
#define TSI_GENCS_DVOLT(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_DVOLT_SHIFT))&TSI_GENCS_DVOLT_MASK)

#define TSI_GENCS_REFCHRG_MASK        0xE000000u
#define TSI_GENCS_REFCHRG_SHIFT       21
#define TSI_GENCS_REFCHRG_WIDTH       3
#define TSI_GENCS_REFCHRG(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_REFCHRG_SHIFT))&TSI_GENCS_REFCHRG_MASK)

#define TSI_GENCS_MODE_MASK           0xF000000u
#define TSI_GENCS_MODE_SHIFT          24
#define TSI_GENCS_MODE_WIDTH          4
#define TSI_GENCS_MODE(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_MODE_SHIFT))&TSI_GENCS_MODE_MASK)

#define TSI_GENCS_ESOR_MASK           0x10000000u
#define TSI_GENCS_ESOR_SHIFT          28

```

```

#define TSI_GENCS_ESOR_WIDTH          1
#define TSI_GENCS_ESOR(x)             (((uint32_t)((uint32_t)(x)<<TSI_GENCS_ESOR_SHIFT))&TSI_GENCS_ESOR_MASK)

#define TSI_GENCS_OUTRGF_MASK         0x80000000u
#define TSI_GENCS_OUTRGF_SHIFT        31
#define TSI_GENCS_OUTRGF_WIDTH        1
#define TSI_GENCS_OUTRGF(x)           (((uint32_t)((uint32_t)(x)<<TSI_GENCS_OUTRGF_SHIFT))&TSI_GENCS_OUTRGF_MASK)

/* DATA Bit Fields */

#define TSI_DATA_TSICNT_MASK          0xFFFFu
#define TSI_DATA_TSICNT_SHIFT         0
#define TSI_DATA_TSICNT_WIDTH         16
#define TSI_DATA_TSICNT(x)            (((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICNT_SHIFT))&TSI_DATA_TSICNT_MASK)

#define TSI_DATA_SWTS_MASK            0x400000u
#define TSI_DATA_SWTS_SHIFT           22
#define TSI_DATA_SWTS_WIDTH           1
#define TSI_DATA_SWTS(x)              (((uint32_t)((uint32_t)(x)<<TSI_DATA_SWTS_SHIFT))&TSI_DATA_SWTS_MASK)

#define TSI_DATA_DMAEN_MASK           0x800000u
#define TSI_DATA_DMAEN_SHIFT          23
#define TSI_DATA_DMAEN_WIDTH          1
#define TSI_DATA_DMAEN(x)             (((uint32_t)((uint32_t)(x)<<TSI_DATA_DMAEN_SHIFT))&TSI_DATA_DMAEN_MASK)

#define TSI_DATA_TSICH_MASK           0xF0000000u
#define TSI_DATA_TSICH_SHIFT          28
#define TSI_DATA_TSICH_WIDTH          4
#define TSI_DATA_TSICH(x)             (((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICH_SHIFT))&TSI_DATA_TSICH_MASK)

/* TSHD Bit Fields */

#define TSI_TSHD_THRESL_MASK          0xFFFFu

```

```

#define TSI_TS HD_THRESL_SHIFT          0
#define TSI_TS HD_THRESL_WIDTH          16
#define TSI_TS HD_THRESL(x)
(((uint32_t)((uint32_t)(x)<<TSI_TS HD_THRESL_SHIFT))&TSI_TS HD_THRESL_MASK)

#define TSI_TS HD_THRESH_MASK           0xFFFF0000u
#define TSI_TS HD_THRESH_SHIFT          16
#define TSI_TS HD_THRESH_WIDTH          16
#define TSI_TS HD_THRESH(x)
(((uint32_t)((uint32_t)(x)<<TSI_TS HD_THRESH_SHIFT))&TSI_TS HD_THRESH_MASK)

/*
 * @}
 */
/* end of group TSI_Register_Masks */

```

```

/* TSI - Peripheral instance base addresses */

/** Peripheral TSI0 base address */
#define TSI0_BASE                  (0x40045000u)

/** Peripheral TSI0 base pointer */
#define TSI0                      ((TSI_Type *)TSI0_BASE)
#define TSI0_BASE_PTR              (TSI0)

/** Array initializer of TSI peripheral base addresses */
#define TSI_BASE_ADDRS             { TSI0_BASE }

/** Array initializer of TSI peripheral base pointers */
#define TSI_BASE_PTRS              { TSI0 }

```

```

/*
-----*
-- TSI - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros  
 * @{  
 */
```

```
/* TSI - Register instance definitions */  
/* TSI0 */  
  
#define TSI0_GENCS          TSI_GENCS_REG(TSI0)  
#define TSI0_DATA           TSI_DATA_REG(TSI0)  
#define TSI0_TS HD          TSI_TS HD_REG(TSI0)
```

```
/*!  
 * @}  
 */ /* end of group TSI_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group TSI_Peripheral_Access_Layer */
```

```
/* -----  
-- UART Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup UART_Peripheral_Access_Layer UART Peripheral Access Layer
```

```

* @@
*/

```

```

/** UART - Register Layout Typedef */

typedef struct {

    __IO uint8_t BDH;           /**< UART Baud Rate Register: High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register: Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __I  uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;              /**< UART Data Register, offset: 0x7 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0x8 */

} UART_Type, *UART_MemMapPtr;

```

```

/* -----
-- UART - Register accessor macros
----- */

```

```

/*!
* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros
* @@
*/
```

```

/* UART - Register accessors */

#define UART_BDH_REG(base)      ((base)->BDH)
#define UART_BDL_REG(base)      ((base)->BDL)
```

```

#define UART_C1_REG(base)          ((base)->C1)
#define UART_C2_REG(base)          ((base)->C2)
#define UART_S1_REG(base)          ((base)->S1)
#define UART_S2_REG(base)          ((base)->S2)
#define UART_C3_REG(base)          ((base)->C3)
#define UART_D_REG(base)           ((base)->D)
#define UART_C4_REG(base)          ((base)->C4)

/*!
 * @}
 */

/* /* end of group UART_Register_Accessor_Macros */

/* -----
-- UART Register Masks
----- */

/*!

 * @addtogroup UART_Register_Masks UART Register Masks
 * @{
 */

/* BDH Bit Fields */

#define UART_BDH_SBR_MASK          0x1Fu
#define UART_BDH_SBR_SHIFT         0
#define UART_BDH_SBR_WIDTH         5
#define UART_BDH_SBR(x)            (((uint8_t)((uint8_t)(x)<<UART_BDH_SBR_SHIFT))&UART_BDH_SBR_MASK)
#define UART_BDH_SBNS_MASK          0x20u

```

```

#define UART_BDH_SBNS_SHIFT      5
#define UART_BDH_SBNS_WIDTH      1
#define UART_BDH_SBNS(x)
(((uint8_t)((uint8_t)(x)<<UART_BDH_SBNS_SHIFT))&UART_BDH_SBNS_MASK)

#define UART_BDH_RXEDGIE_MASK    0x40u
#define UART_BDH_RXEDGIE_SHIFT   6
#define UART_BDH_RXEDGIE_WIDTH   1
#define UART_BDH_RXEDGIE(x)
(((uint8_t)((uint8_t)(x)<<UART_BDH_RXEDGIE_SHIFT))&UART_BDH_RXEDGIE_MASK)

#define UART_BDH_LBKDIE_MASK     0x80u
#define UART_BDH_LBKDIE_SHIFT    7
#define UART_BDH_LBKDIE_WIDTH    1
#define UART_BDH_LBKDIE(x)
(((uint8_t)((uint8_t)(x)<<UART_BDH_LBKDIE_SHIFT))&UART_BDH_LBKDIE_MASK)

/* BDL Bit Fields */

#define UART_BDL_SBR_MASK        0xFFu
#define UART_BDL_SBR_SHIFT       0
#define UART_BDL_SBR_WIDTH       8
#define UART_BDL_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART_BDL_SBR_SHIFT))&UART_BDL_SBR_MASK)

/* C1 Bit Fields */

#define UART_C1_PT_MASK          0x1u
#define UART_C1_PT_SHIFT         0
#define UART_C1_PT_WIDTH         1
#define UART_C1_PT(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_PT_SHIFT))&UART_C1_PT_MASK)

#define UART_C1_PE_MASK          0x2u
#define UART_C1_PE_SHIFT         1
#define UART_C1_PE_WIDTH         1
#define UART_C1_PE(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_PE_SHIFT))&UART_C1_PE_MASK)

```

```

#define UART_C1_ILT_MASK           0x4u
#define UART_C1_ILT_SHIFT          2
#define UART_C1_ILT_WIDTH          1
#define UART_C1_ILT(x)             (((uint8_t)((uint8_t)(x))<<UART_C1_ILT_SHIFT))&UART_C1_ILT_MASK

#define UART_C1_WAKE_MASK          0x8u
#define UART_C1_WAKE_SHIFT         3
#define UART_C1_WAKE_WIDTH         1
#define UART_C1_WAKE(x)            (((uint8_t)((uint8_t)(x))<<UART_C1_WAKE_SHIFT))&UART_C1_WAKE_MASK

#define UART_C1_M_MASK             0x10u
#define UART_C1_M_SHIFT            4
#define UART_C1_M_WIDTH            1
#define UART_C1_M(x)               (((uint8_t)((uint8_t)(x))<<UART_C1_M_SHIFT))&UART_C1_M_MASK

#define UART_C1_RSRC_MASK          0x20u
#define UART_C1_RSRC_SHIFT         5
#define UART_C1_RSRC_WIDTH         1
#define UART_C1_RSRC(x)            (((uint8_t)((uint8_t)(x))<<UART_C1_RSRC_SHIFT))&UART_C1_RSRC_MASK

#define UART_C1_UARTSWAI_MASK       0x40u
#define UART_C1_UARTSWAI_SHIFT      6
#define UART_C1_UARTSWAI_WIDTH      1
#define UART_C1_UARTSWAI(x)         (((uint8_t)((uint8_t)(x))<<UART_C1_UARTSWAI_SHIFT))&UART_C1_UARTSWAI_MASK

#define UART_C1_LOOPS_MASK          0x80u
#define UART_C1_LOOPS_SHIFT         7
#define UART_C1_LOOPS_WIDTH         1
#define UART_C1_LOOPS(x)            (((uint8_t)((uint8_t)(x))<<UART_C1_LOOPS_SHIFT))&UART_C1_LOOPS_MASK

/* C2 Bit Fields */

```

```

#define UART_C2_SBK_MASK           0x1u
#define UART_C2_SBK_SHIFT          0
#define UART_C2_SBK_WIDTH          1
#define UART_C2_SBK(x) (((uint8_t)((uint8_t)(x))<<UART_C2_SBK_SHIFT))&UART_C2_SBK_MASK

#define UART_C2_RWU_MASK           0x2u
#define UART_C2_RWU_SHIFT          1
#define UART_C2_RWU_WIDTH          1
#define UART_C2_RWU(x) (((uint8_t)((uint8_t)(x))<<UART_C2_RWU_SHIFT))&UART_C2_RWU_MASK

#define UART_C2_RE_MASK            0x4u
#define UART_C2_RE_SHIFT           2
#define UART_C2_RE_WIDTH           1
#define UART_C2_RE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_RE_SHIFT))&UART_C2_RE_MASK

#define UART_C2_TE_MASK            0x8u
#define UART_C2_TE_SHIFT           3
#define UART_C2_TE_WIDTH           1
#define UART_C2_TE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_TE_SHIFT))&UART_C2_TE_MASK

#define UART_C2_ILIE_MASK          0x10u
#define UART_C2_ILIE_SHIFT          4
#define UART_C2_ILIE_WIDTH          1
#define UART_C2_ILIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_ILIE_SHIFT))&UART_C2_ILIE_MASK

#define UART_C2_RIE_MASK           0x20u
#define UART_C2_RIE_SHIFT          5
#define UART_C2_RIE_WIDTH           1
#define UART_C2_RIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_RIE_SHIFT))&UART_C2_RIE_MASK

#define UART_C2_TCIE_MASK          0x40u

```

```

#define UART_C2_TCIE_SHIFT          6
#define UART_C2_TCIE_WIDTH          1
#define UART_C2_TCIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TCIE_SHIFT))&UART_C2_TCIE_MASK)

#define UART_C2_TIE_MASK            0x80u
#define UART_C2_TIE_SHIFT           7
#define UART_C2_TIE_WIDTH           1
#define UART_C2_TIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TIE_SHIFT))&UART_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART_S1_PF_MASK             0x1u
#define UART_S1_PF_SHIFT             0
#define UART_S1_PF_WIDTH             1
#define UART_S1_PF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_PF_SHIFT))&UART_S1_PF_MASK)

#define UART_S1_FE_MASK              0x2u
#define UART_S1_FE_SHIFT              1
#define UART_S1_FE_WIDTH              1
#define UART_S1_FE(x) (((uint8_t)((uint8_t)(x)<<UART_S1_FE_SHIFT))&UART_S1_FE_MASK)

#define UART_S1_NF_MASK              0x4u
#define UART_S1_NF_SHIFT              2
#define UART_S1_NF_WIDTH              1
#define UART_S1_NF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_NF_SHIFT))&UART_S1_NF_MASK)

#define UART_S1_OR_MASK              0x8u
#define UART_S1_OR_SHIFT              3
#define UART_S1_OR_WIDTH              1
#define UART_S1_OR(x) (((uint8_t)((uint8_t)(x)<<UART_S1_OR_SHIFT))&UART_S1_OR_MASK)

#define UART_S1_IDLE_MASK             0x10u

```

```

#define UART_S1_IDLE_SHIFT          4
#define UART_S1_IDLE_WIDTH          1
#define UART_S1_IDLE(x)             (((uint8_t)((uint8_t)(x)<<UART_S1_IDLE_SHIFT))&UART_S1_IDLE_MASK)

#define UART_S1_RDRF_MASK           0x20u
#define UART_S1_RDRF_SHIFT          5
#define UART_S1_RDRF_WIDTH          1
#define UART_S1_RDRF(x)             (((uint8_t)((uint8_t)(x)<<UART_S1_RDRF_SHIFT))&UART_S1_RDRF_MASK)

#define UART_S1_TC_MASK              0x40u
#define UART_S1_TC_SHIFT             6
#define UART_S1_TC_WIDTH             1
#define UART_S1_TC(x)                (((uint8_t)((uint8_t)(x)<<UART_S1_TC_SHIFT))&UART_S1_TC_MASK)

#define UART_S1_TDRE_MASK            0x80u
#define UART_S1_TDRE_SHIFT           7
#define UART_S1_TDRE_WIDTH           1
#define UART_S1_TDRE(x)               (((uint8_t)((uint8_t)(x)<<UART_S1_TDRE_SHIFT))&UART_S1_TDRE_MASK)

/* S2 Bit Fields */

#define UART_S2_RAF_MASK              0x1u
#define UART_S2_RAF_SHIFT              0
#define UART_S2_RAF_WIDTH              1
#define UART_S2_RAF(x)                 (((uint8_t)((uint8_t)(x)<<UART_S2_RAF_SHIFT))&UART_S2_RAF_MASK)

#define UART_S2_LBKDE_MASK             0x2u
#define UART_S2_LBKDE_SHIFT            1
#define UART_S2_LBKDE_WIDTH            1
#define UART_S2_LBKDE(x)                (((uint8_t)((uint8_t)(x)<<UART_S2_LBKDE_SHIFT))&UART_S2_LBKDE_MASK)

#define UART_S2_BRK13_MASK             0x4u

```

```

#define UART_S2_BRK13_SHIFT          2
#define UART_S2_BRK13_WIDTH          1
#define UART_S2_BRK13(x)
(((uint8_t)((uint8_t)(x)<<UART_S2_BRK13_SHIFT))&UART_S2_BRK13_MASK)

#define UART_S2_RWUID_MASK           0x8u
#define UART_S2_RWUID_SHIFT          3
#define UART_S2_RWUID_WIDTH          1
#define UART_S2_RWUID(x)
(((uint8_t)((uint8_t)(x)<<UART_S2_RWUID_SHIFT))&UART_S2_RWUID_MASK)

#define UART_S2_RXINV_MASK           0x10u
#define UART_S2_RXINV_SHIFT          4
#define UART_S2_RXINV_WIDTH          1
#define UART_S2_RXINV(x)
(((uint8_t)((uint8_t)(x)<<UART_S2_RXINV_SHIFT))&UART_S2_RXINV_MASK)

#define UART_S2_RXEDGIF_MASK          0x40u
#define UART_S2_RXEDGIF_SHIFT         6
#define UART_S2_RXEDGIF_WIDTH         1
#define UART_S2_RXEDGIF(x)
(((uint8_t)((uint8_t)(x)<<UART_S2_RXEDGIF_SHIFT))&UART_S2_RXEDGIF_MASK)

#define UART_S2_LBKDIF_MASK           0x80u
#define UART_S2_LBKDIF_SHIFT          7
#define UART_S2_LBKDIF_WIDTH          1
#define UART_S2_LBKDIF(x)
(((uint8_t)((uint8_t)(x)<<UART_S2_LBKDIF_SHIFT))&UART_S2_LBKDIF_MASK)

/* C3 Bit Fields */

#define UART_C3_PEIE_MASK             0x1u
#define UART_C3_PEIE_SHIFT             0
#define UART_C3_PEIE_WIDTH             1
#define UART_C3_PEIE(x)
(((uint8_t)((uint8_t)(x)<<UART_C3_PEIE_SHIFT))&UART_C3_PEIE_MASK)

#define UART_C3_FEIE_MASK             0x2u

```

```

#define UART_C3_FEIE_SHIFT          1
#define UART_C3_FEIE_WIDTH          1
#define UART_C3_FEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_FEIE_SHIFT))&UART_C3_FEIE_MASK)

#define UART_C3_NEIE_MASK           0x4u
#define UART_C3_NEIE_SHIFT          2
#define UART_C3_NEIE_WIDTH          1
#define UART_C3_NEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_NEIE_SHIFT))&UART_C3_NEIE_MASK)

#define UART_C3_ORIE_MASK           0x8u
#define UART_C3_ORIE_SHIFT          3
#define UART_C3_ORIE_WIDTH          1
#define UART_C3_ORIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_ORIE_SHIFT))&UART_C3_ORIE_MASK)

#define UART_C3_TXINV_MASK           0x10u
#define UART_C3_TXINV_SHIFT          4
#define UART_C3_TXINV_WIDTH          1
#define UART_C3_TXINV(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXINV_SHIFT))&UART_C3_TXINV_MASK)

#define UART_C3_TXDIR_MASK           0x20u
#define UART_C3_TXDIR_SHIFT          5
#define UART_C3_TXDIR_WIDTH          1
#define UART_C3_TXDIR(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXDIR_SHIFT))&UART_C3_TXDIR_MASK)

#define UART_C3_T8_MASK              0x40u
#define UART_C3_T8_SHIFT              6
#define UART_C3_T8_WIDTH              1
#define UART_C3_T8(x) (((uint8_t)((uint8_t)(x)<<UART_C3_T8_SHIFT))&UART_C3_T8_MASK)

#define UART_C3_R8_MASK              0x80u
#define UART_C3_R8_SHIFT              7

```

```

#define UART_C3_R8_WIDTH          1

#define UART_C3_R8(x) \
(((uint8_t)((uint8_t)(x)<<UART_C3_R8_SHIFT))&UART_C3_R8_MASK)

/* D Bit Fields */

#define UART_D_R0T0_MASK          0x1u
#define UART_D_R0T0_SHIFT         0
#define UART_D_R0T0_WIDTH          1

#define UART_D_R0T0(x) \
(((uint8_t)((uint8_t)(x)<<UART_D_R0T0_SHIFT))&UART_D_R0T0_MASK)

#define UART_D_R1T1_MASK          0x2u
#define UART_D_R1T1_SHIFT         1
#define UART_D_R1T1_WIDTH          1

#define UART_D_R1T1(x) \
(((uint8_t)((uint8_t)(x)<<UART_D_R1T1_SHIFT))&UART_D_R1T1_MASK)

#define UART_D_R2T2_MASK          0x4u
#define UART_D_R2T2_SHIFT         2
#define UART_D_R2T2_WIDTH          1

#define UART_D_R2T2(x) \
(((uint8_t)((uint8_t)(x)<<UART_D_R2T2_SHIFT))&UART_D_R2T2_MASK)

#define UART_D_R3T3_MASK          0x8u
#define UART_D_R3T3_SHIFT         3
#define UART_D_R3T3_WIDTH          1

#define UART_D_R3T3(x) \
(((uint8_t)((uint8_t)(x)<<UART_D_R3T3_SHIFT))&UART_D_R3T3_MASK)

#define UART_D_R4T4_MASK          0x10u
#define UART_D_R4T4_SHIFT         4
#define UART_D_R4T4_WIDTH          1

#define UART_D_R4T4(x) \
(((uint8_t)((uint8_t)(x)<<UART_D_R4T4_SHIFT))&UART_D_R4T4_MASK)

#define UART_D_R5T5_MASK          0x20u
#define UART_D_R5T5_SHIFT         5

```

```

#define UART_D_R5T5_WIDTH          1
#define UART_D_R5T5(x) (((uint8_t)((uint8_t)(x)<<UART_D_R5T5_SHIFT))&UART_D_R5T5_MASK)
#define UART_D_R6T6_MASK           0x40u
#define UART_D_R6T6_SHIFT          6
#define UART_D_R6T6_WIDTH          1
#define UART_D_R6T6(x) (((uint8_t)((uint8_t)(x)<<UART_D_R6T6_SHIFT))&UART_D_R6T6_MASK)
#define UART_D_R7T7_MASK           0x80u
#define UART_D_R7T7_SHIFT          7
#define UART_D_R7T7_WIDTH          1
#define UART_D_R7T7(x) (((uint8_t)((uint8_t)(x)<<UART_D_R7T7_SHIFT))&UART_D_R7T7_MASK)

/* C4 Bit Fields */

#define UART_C4_RDMAS_MASK         0x20u
#define UART_C4_RDMAS_SHIFT         5
#define UART_C4_RDMAS_WIDTH         1
#define UART_C4_RDMAS(x) (((uint8_t)((uint8_t)(x)<<UART_C4_RDMAS_SHIFT))&UART_C4_RDMAS_MASK)
#define UART_C4_TDMAS_MASK          0x80u
#define UART_C4_TDMAS_SHIFT         7
#define UART_C4_TDMAS_WIDTH         1
#define UART_C4_TDMAS(x) (((uint8_t)((uint8_t)(x)<<UART_C4_TDMAS_SHIFT))&UART_C4_TDMAS_MASK)

/*
 * @}
 */ /* end of group UART_Register_Masks */

```

```

/* UART - Peripheral instance base addresses */

/** Peripheral UART1 base address */

#define UART1_BASE          (0x4006B000u)

/** Peripheral UART1 base pointer */

#define UART1           ((UART_Type *)UART1_BASE)

#define UART1_BASE_PTR      (UART1)

/** Peripheral UART2 base address */

#define UART2_BASE          (0x4006C000u)

/** Peripheral UART2 base pointer */

#define UART2           ((UART_Type *)UART2_BASE)

#define UART2_BASE_PTR      (UART2)

/** Array initializer of UART peripheral base addresses */

#define UART_BASE_ADDRS     { UART1_BASE, UART2_BASE }

/** Array initializer of UART peripheral base pointers */

#define UART_BASE_PTRS       { UART1, UART2 }

/*
-----*
-- UART - Register accessor macros
----- */

```

```

/*!

* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros
* @{
* /

```

```

/* UART - Register instance definitions */

/* UART1 */

#define UART1_BDH          UART_BDH_REG(UART1)

```

```
#define UART1_BDL          UART_BDL_REG(UART1)
#define UART1_C1           UART_C1_REG(UART1)
#define UART1_C2           UART_C2_REG(UART1)
#define UART1_S1           UART_S1_REG(UART1)
#define UART1_S2           UART_S2_REG(UART1)
#define UART1_C3           UART_C3_REG(UART1)
#define UART1_D            UART_D_REG(UART1)
#define UART1_C4           UART_C4_REG(UART1)

/* UART2 */

#define UART2_BDH          UART_BDH_REG(UART2)
#define UART2_BDL          UART_BDL_REG(UART2)
#define UART2_C1           UART_C1_REG(UART2)
#define UART2_C2           UART_C2_REG(UART2)
#define UART2_S1           UART_S1_REG(UART2)
#define UART2_S2           UART_S2_REG(UART2)
#define UART2_C3           UART_C3_REG(UART2)
#define UART2_D            UART_D_REG(UART2)
#define UART2_C4           UART_C4_REG(UART2)

/*!
 * @}
 */
/* /* end of group UART_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* /* end of group UART_Peripheral_Access_Layer */
```

```

/*
-- UART0 Peripheral Access Layer
----- */
/*!
* @addtogroup UART0_Peripheral_Access_Layer UART0 Peripheral Access Layer
* @{
*/
/** UART0 - Register Layout Typedef */
typedef struct {
    __IO uint8_t BDH;           /**< UART Baud Rate Register High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __IO uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;              /**< UART Data Register, offset: 0x7 */
    __IO uint8_t MA1;           /**< UART Match Address Registers 1, offset: 0x8 */
    __IO uint8_t MA2;           /**< UART Match Address Registers 2, offset: 0x9 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0xA */
    __IO uint8_t C5;            /**< UART Control Register 5, offset: 0xB */
} UART0_Type, *UART0_MemMapPtr;

/*
-- UART0 - Register accessor macros
----- */

```

```

/*!

* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros

* @{

*/



/* UART0 - Register accessors */

#define UART0_BDH_REG(base)          ((base)->BDH)
#define UART0_BDL_REG(base)          ((base)->BDL)
#define UART0_C1_REG(base)           ((base)->C1)
#define UART0_C2_REG(base)           ((base)->C2)
#define UART0_S1_REG(base)           ((base)->S1)
#define UART0_S2_REG(base)           ((base)->S2)
#define UART0_C3_REG(base)           ((base)->C3)
#define UART0_D_REG(base)            ((base)->D)
#define UART0_MA1_REG(base)          ((base)->MA1)
#define UART0_MA2_REG(base)          ((base)->MA2)
#define UART0_C4_REG(base)           ((base)->C4)
#define UART0_C5_REG(base)           ((base)->C5)

!



/* -----
-- UART0 Register Masks
----- */

```

```

/*!

* @addtogroup UART0_Register_Masks UART0 Register Masks

* @{

*/



/* BDH Bit Fields */

#define UART0_BDH_SBR_MASK          0x1Fu
#define UART0_BDH_SBR_SHIFT         0
#define UART0_BDH_SBR_WIDTH         5
#define UART0_BDH_SBR(x)            (((uint8_t)((uint8_t)(x))<<UART0_BDH_SBR_SHIFT))&UART0_BDH_SBR_MASK)

#define UART0_BDH_SBNS_MASK         0x20u
#define UART0_BDH_SBNS_SHIFT        5
#define UART0_BDH_SBNS_WIDTH        1
#define UART0_BDH_SBNS(x)           (((uint8_t)((uint8_t)(x))<<UART0_BDH_SBNS_SHIFT))&UART0_BDH_SBNS_MASK)

#define UART0_BDH_RXEDGIE_MASK      0x40u
#define UART0_BDH_RXEDGIE_SHIFT     6
#define UART0_BDH_RXEDGIE_WIDTH     1
#define UART0_BDH_RXEDGIE(x)        (((uint8_t)((uint8_t)(x))<<UART0_BDH_RXEDGIE_SHIFT))&UART0_BDH_RXEDGIE_MASK)

#define UART0_BDH_LBKDIE_MASK        0x80u
#define UART0_BDH_LBKDIE_SHIFT       7
#define UART0_BDH_LBKDIE_WIDTH       1
#define UART0_BDH_LBKDIE(x)         (((uint8_t)((uint8_t)(x))<<UART0_BDH_LBKDIE_SHIFT))&UART0_BDH_LBKDIE_MASK)

/* BDL Bit Fields */

#define UART0_BDL_SBR_MASK          0xFFu
#define UART0_BDL_SBR_SHIFT         0
#define UART0_BDL_SBR_WIDTH         8

```

```

#define UART0_BDL_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART0_BDL_SBR_SHIFT))&UART0_BDL_SBR_MASK)

/* C1 Bit Fields */

#define UART0_C1_PT_MASK          0x1u
#define UART0_C1_PT_SHIFT         0
#define UART0_C1_PT_WIDTH         1

#define UART0_C1_PT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PT_SHIFT))&UART0_C1_PT_MASK)

#define UART0_C1_PE_MASK          0x2u
#define UART0_C1_PE_SHIFT         1
#define UART0_C1_PE_WIDTH         1

#define UART0_C1_PE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PE_SHIFT))&UART0_C1_PE_MASK)

#define UART0_C1_ILT_MASK          0x4u
#define UART0_C1_ILT_SHIFT         2
#define UART0_C1_ILT_WIDTH         1

#define UART0_C1_ILT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_ILT_SHIFT))&UART0_C1_ILT_MASK)

#define UART0_C1_WAKE_MASK          0x8u
#define UART0_C1_WAKE_SHIFT         3
#define UART0_C1_WAKE_WIDTH         1

#define UART0_C1_WAKE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_WAKE_SHIFT))&UART0_C1_WAKE_MASK)

#define UART0_C1_M_MASK           0x10u
#define UART0_C1_M_SHIFT            4
#define UART0_C1_M_WIDTH             1

#define UART0_C1_M(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_M_SHIFT))&UART0_C1_M_MASK)

#define UART0_C1_RSRC_MASK          0x20u
#define UART0_C1_RSRC_SHIFT         5
#define UART0_C1_RSRC_WIDTH         1

```

```

#define UART0_C1_RSRC(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_RSRC_SHIFT))&UART0_C1_RSRC_MASK)

#define UART0_C1_DOZEEN_MASK          0x40u
#define UART0_C1_DOZEEN_SHIFT        6
#define UART0_C1_DOZEEN_WIDTH        1

#define UART0_C1_DOZEEN(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_DOZEEN_SHIFT))&UART0_C1_DOZEEN_MASK)

#define UART0_C1_LOOPS_MASK          0x80u
#define UART0_C1_LOOPS_SHIFT        7
#define UART0_C1_LOOPS_WIDTH        1

#define UART0_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_LOOPS_SHIFT))&UART0_C1_LOOPS_MASK)

/* C2 Bit Fields */

#define UART0_C2_SBK_MASK           0x1u
#define UART0_C2_SBK_SHIFT          0
#define UART0_C2_SBK_WIDTH          1

#define UART0_C2_SBK(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_SBK_SHIFT))&UART0_C2_SBK_MASK)

#define UART0_C2_RWU_MASK           0x2u
#define UART0_C2_RWU_SHIFT          1
#define UART0_C2_RWU_WIDTH          1

#define UART0_C2_RWU(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RWU_SHIFT))&UART0_C2_RWU_MASK)

#define UART0_C2_RE_MASK            0x4u
#define UART0_C2_RE_SHIFT           2
#define UART0_C2_RE_WIDTH           1

#define UART0_C2_RE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RE_SHIFT))&UART0_C2_RE_MASK)

#define UART0_C2_TE_MASK            0x8u
#define UART0_C2_TE_SHIFT           3
#define UART0_C2_TE_WIDTH           1

```

```

#define UART0_C2_TE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TE_SHIFT))&UART0_C2_TE_MASK)

#define UART0_C2_ILIE_MASK          0x10u
#define UART0_C2_ILIE_SHIFT         4
#define UART0_C2_ILIE_WIDTH         1

#define UART0_C2_ILIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_ILIE_SHIFT))&UART0_C2_ILIE_MASK)

#define UART0_C2_RIE_MASK           0x20u
#define UART0_C2_RIE_SHIFT          5
#define UART0_C2_RIE_WIDTH          1

#define UART0_C2_RIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_RIE_SHIFT))&UART0_C2_RIE_MASK)

#define UART0_C2_TCIE_MASK          0x40u
#define UART0_C2_TCIE_SHIFT          6
#define UART0_C2_TCIE_WIDTH          1

#define UART0_C2_TCIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TCIE_SHIFT))&UART0_C2_TCIE_MASK)

#define UART0_C2_TIE_MASK            0x80u
#define UART0_C2_TIE_SHIFT           7
#define UART0_C2_TIE_WIDTH           1

#define UART0_C2_TIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TIE_SHIFT))&UART0_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART0_S1_PF_MASK             0x1u
#define UART0_S1_PF_SHIFT              0
#define UART0_S1_PF_WIDTH              1

#define UART0_S1_PF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_PF_SHIFT))&UART0_S1_PF_MASK)

#define UART0_S1_FE_MASK              0x2u
#define UART0_S1_FE_SHIFT                1
#define UART0_S1_FE_WIDTH                1

```

```

#define UART0_S1_FE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_FE_SHIFT))&UART0_S1_FE_MASK)

#define UART0_S1_NF_MASK          0x4u
#define UART0_S1_NF_SHIFT         2
#define UART0_S1_NF_WIDTH         1

#define UART0_S1_NF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_NF_SHIFT))&UART0_S1_NF_MASK)

#define UART0_S1_OR_MASK          0x8u
#define UART0_S1_OR_SHIFT         3
#define UART0_S1_OR_WIDTH         1

#define UART0_S1_OR(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_OR_SHIFT))&UART0_S1_OR_MASK)

#define UART0_S1_IDLE_MASK         0x10u
#define UART0_S1_IDLE_SHIFT        4
#define UART0_S1_IDLE_WIDTH        1

#define UART0_S1_IDLE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_IDLE_SHIFT))&UART0_S1_IDLE_MASK)

#define UART0_S1_RDRF_MASK         0x20u
#define UART0_S1_RDRF_SHIFT        5
#define UART0_S1_RDRF_WIDTH        1

#define UART0_S1_RDRF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_RDRF_SHIFT))&UART0_S1_RDRF_MASK)

#define UART0_S1_TC_MASK           0x40u
#define UART0_S1_TC_SHIFT          6
#define UART0_S1_TC_WIDTH          1

#define UART0_S1_TC(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TC_SHIFT))&UART0_S1_TC_MASK)

#define UART0_S1_TDRE_MASK          0x80u
#define UART0_S1_TDRE_SHIFT         7
#define UART0_S1_TDRE_WIDTH         1

```

```

#define UART0_S1_TDRE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TDRE_SHIFT))&UART0_S1_TDRE_MASK)

/* S2 Bit Fields */

#define UART0_S2_RAF_MASK          0x1u
#define UART0_S2_RAF_SHIFT         0
#define UART0_S2_RAF_WIDTH         1

#define UART0_S2_RAF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RAF_SHIFT))&UART0_S2_RAF_MASK)

#define UART0_S2_LBKDE_MASK        0x2u
#define UART0_S2_LBKDE_SHIFT       1
#define UART0_S2_LBKDE_WIDTH       1

#define UART0_S2_LBKDE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_LBKDE_SHIFT))&UART0_S2_LBKDE_MASK)

#define UART0_S2_BRK13_MASK         0x4u
#define UART0_S2_BRK13_SHIFT        2
#define UART0_S2_BRK13_WIDTH        1

#define UART0_S2_BRK13(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_BRK13_SHIFT))&UART0_S2_BRK13_MASK)

#define UART0_S2_RWUID_MASK         0x8u
#define UART0_S2_RWUID_SHIFT        3
#define UART0_S2_RWUID_WIDTH        1

#define UART0_S2_RWUID(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RWUID_SHIFT))&UART0_S2_RWUID_MASK)

#define UART0_S2_RXINV_MASK         0x10u
#define UART0_S2_RXINV_SHIFT        4
#define UART0_S2_RXINV_WIDTH        1

#define UART0_S2_RXINV(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RXINV_SHIFT))&UART0_S2_RXINV_MASK)

#define UART0_S2_MSBF_MASK          0x20u
#define UART0_S2_MSBF_SHIFT         5
#define UART0_S2_MSBF_WIDTH         1

```

```

#define UART0_S2_MSBF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_MSBF_SHIFT))&UART0_S2_MSBF_MASK

#define UART0_S2_RXEDGIF_MASK          0x40u
#define UART0_S2_RXEDGIF_SHIFT        6
#define UART0_S2_RXEDGIF_WIDTH        1

#define UART0_S2_RXEDGIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_RXEDGIF_SHIFT))&UART0_S2_RXEDGIF_MASK

#define UART0_S2_LBKDIF_MASK          0x80u
#define UART0_S2_LBKDIF_SHIFT        7
#define UART0_S2_LBKDIF_WIDTH        1

#define UART0_S2_LBKDIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_LBKDIF_SHIFT))&UART0_S2_LBKDIF_MASK

/* C3 Bit Fields */

#define UART0_C3_PEIE_MASK           0x1u
#define UART0_C3_PEIE_SHIFT          0
#define UART0_C3_PEIE_WIDTH          1

#define UART0_C3_PEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_PEIE_SHIFT))&UART0_C3_PEIE_MASK

#define UART0_C3_FEIE_MASK           0x2u
#define UART0_C3_FEIE_SHIFT          1
#define UART0_C3_FEIE_WIDTH          1

#define UART0_C3_FEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_FEIE_SHIFT))&UART0_C3_FEIE_MASK

#define UART0_C3_NEIE_MASK           0x4u
#define UART0_C3_NEIE_SHIFT          2
#define UART0_C3_NEIE_WIDTH          1

#define UART0_C3_NEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_NEIE_SHIFT))&UART0_C3_NEIE_MASK

#define UART0_C3_ORIE_MASK           0x8u
#define UART0_C3_ORIE_SHIFT          3
#define UART0_C3_ORIE_WIDTH          1

```

```

#define UART0_C3_ORIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_ORIE_SHIFT))&UART0_C3_ORIE_MASK)

#define UART0_C3_TXINV_MASK          0x10u
#define UART0_C3_TXINV_SHIFT        4
#define UART0_C3_TXINV_WIDTH        1

#define UART0_C3_TXINV(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_TXINV_SHIFT))&UART0_C3_TXINV_MASK)

#define UART0_C3_TXDIR_MASK          0x20u
#define UART0_C3_TXDIR_SHIFT        5
#define UART0_C3_TXDIR_WIDTH        1

#define UART0_C3_TXDIR(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_TXDIR_SHIFT))&UART0_C3_TXDIR_MASK)

#define UART0_C3_R9T8_MASK          0x40u
#define UART0_C3_R9T8_SHIFT         6
#define UART0_C3_R9T8_WIDTH         1

#define UART0_C3_R9T8(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_R9T8_SHIFT))&UART0_C3_R9T8_MASK)

#define UART0_C3_R8T9_MASK          0x80u
#define UART0_C3_R8T9_SHIFT         7
#define UART0_C3_R8T9_WIDTH         1

#define UART0_C3_R8T9(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_R8T9_SHIFT))&UART0_C3_R8T9_MASK)

/* D Bit Fields */

#define UART0_D_ROT0_MASK           0x1u
#define UART0_D_ROT0_SHIFT          0
#define UART0_D_ROT0_WIDTH          1

#define UART0_D_ROT0(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_ROT0_SHIFT))&UART0_D_ROT0_MASK)

#define UART0_D_R1T1_MASK           0x2u
#define UART0_D_R1T1_SHIFT          1
#define UART0_D_R1T1_WIDTH          1

```

```

#define UART0_D_R1T1(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R1T1_SHIFT))&UART0_D_R1T1_MASK)

#define UART0_D_R2T2_MASK          0x4u
#define UART0_D_R2T2_SHIFT         2
#define UART0_D_R2T2_WIDTH         1

#define UART0_D_R2T2(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R2T2_SHIFT))&UART0_D_R2T2_MASK)

#define UART0_D_R3T3_MASK          0x8u
#define UART0_D_R3T3_SHIFT         3
#define UART0_D_R3T3_WIDTH         1

#define UART0_D_R3T3(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R3T3_SHIFT))&UART0_D_R3T3_MASK)

#define UART0_D_R4T4_MASK          0x10u
#define UART0_D_R4T4_SHIFT         4
#define UART0_D_R4T4_WIDTH         1

#define UART0_D_R4T4(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R4T4_SHIFT))&UART0_D_R4T4_MASK)

#define UART0_D_R5T5_MASK          0x20u
#define UART0_D_R5T5_SHIFT         5
#define UART0_D_R5T5_WIDTH         1

#define UART0_D_R5T5(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R5T5_SHIFT))&UART0_D_R5T5_MASK)

#define UART0_D_R6T6_MASK          0x40u
#define UART0_D_R6T6_SHIFT         6
#define UART0_D_R6T6_WIDTH         1

#define UART0_D_R6T6(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R6T6_SHIFT))&UART0_D_R6T6_MASK)

#define UART0_D_R7T7_MASK          0x80u
#define UART0_D_R7T7_SHIFT         7
#define UART0_D_R7T7_WIDTH         1

```

```

#define UART0_D_R7T7(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R7T7_SHIFT))&UART0_D_R7T7_MASK)

/* MA1 Bit Fields */

#define UART0_MA1_MA_MASK          0xFFu
#define UART0_MA1_MA_SHIFT         0
#define UART0_MA1_MA_WIDTH         8

#define UART0_MA1_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA1_MA_SHIFT))&UART0_MA1_MA_MASK)

/* MA2 Bit Fields */

#define UART0_MA2_MA_MASK          0xFFu
#define UART0_MA2_MA_SHIFT         0
#define UART0_MA2_MA_WIDTH         8

#define UART0_MA2_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA2_MA_SHIFT))&UART0_MA2_MA_MASK)

/* C4 Bit Fields */

#define UART0_C4_OSR_MASK          0x1Fu
#define UART0_C4_OSR_SHIFT         0
#define UART0_C4_OSR_WIDTH         5

#define UART0_C4_OSR(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_OSR_SHIFT))&UART0_C4_OSR_MASK)

#define UART0_C4_M10_MASK          0x20u
#define UART0_C4_M10_SHIFT         5
#define UART0_C4_M10_WIDTH         1

#define UART0_C4_M10(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_M10_SHIFT))&UART0_C4_M10_MASK)

#define UART0_C4_MAEN2_MASK         0x40u
#define UART0_C4_MAEN2_SHIFT        6
#define UART0_C4_MAEN2_WIDTH        1

#define UART0_C4_MAEN2(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN2_SHIFT))&UART0_C4_MAEN2_MASK)

#define UART0_C4_MAEN1_MASK         0x80u

```

```

#define UART0_C4_MAEN1_SHIFT          7
#define UART0_C4_MAEN1_WIDTH          1
#define UART0_C4_MAEN1(x) (((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN1_SHIFT))&UART0_C4_MAEN1_MASK)

/* C5 Bit Fields */

#define UART0_C5_RESETDISC_MASK        0x1u
#define UART0_C5_RESETDISC_SHIFT       0
#define UART0_C5_RESETDISC_WIDTH       1
#define UART0_C5_RESETDISC(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RESETDISC_SHIFT))&UART0_C5_RESETDISC_MASK)

#define UART0_C5_BOTHEDGE_MASK         0x2u
#define UART0_C5_BOTHEDGE_SHIFT        1
#define UART0_C5_BOTHEDGE_WIDTH        1
#define UART0_C5_BOTHEDGE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_BOTHEDGE_SHIFT))&UART0_C5_BOTHEDGE_MASK)

#define UART0_C5_RDMAE_MASK            0x20u
#define UART0_C5_RDMAE_SHIFT           5
#define UART0_C5_RDMAE_WIDTH           1
#define UART0_C5_RDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RDMAE_SHIFT))&UART0_C5_RDMAE_MASK)

#define UART0_C5_TDMAE_MASK            0x80u
#define UART0_C5_TDMAE_SHIFT           7
#define UART0_C5_TDMAE_WIDTH           1
#define UART0_C5_TDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_TDMAE_SHIFT))&UART0_C5_TDMAE_MASK)

*/
* @}
/* end of group UART0_Register_Masks */

```

```

/* UART0 - Peripheral instance base addresses */

/** Peripheral UART0 base address */

#define UART0_BASE          (0x4006A000u)

/** Peripheral UART0 base pointer */

#define UART0          ((UART0_Type *)UART0_BASE)

#define UART0_BASE_PTR      (UART0)

/** Array initializer of UART0 peripheral base addresses */

#define UART0_BASE_ADDRS     { UART0_BASE }

/** Array initializer of UART0 peripheral base pointers */

#define UART0_BASE_PTRS      { UART0 }

/*
-----*
-- UART0 - Register accessor macros
----- */
/*!

* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros
* @{
*/
/* UART0 - Register instance definitions */

/* UART0 */

#define UART0_BDH          UART0_BDH_REG(UART0)
#define UART0_BDL          UART0_BDL_REG(UART0)
#define UART0_C1           UART0_C1_REG(UART0)
#define UART0_C2           UART0_C2_REG(UART0)
#define UART0_S1           UART0_S1_REG(UART0)

```

```
#define UART0_S2          UART0_S2_REG(UART0)
#define UART0_C3          UART0_C3_REG(UART0)
#define UART0_D           UART0_D_REG(UART0)
#define UART0_MA1         UART0_MA1_REG(UART0)
#define UART0_MA2         UART0_MA2_REG(UART0)
#define UART0_C4           UART0_C4_REG(UART0)
#define UART0_C5           UART0_C5_REG(UART0)
```

```
/*!
 * @}
 */
/* end of group UART0_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* end of group UART0_Peripheral_Access_Layer */
```

```
/*
-----  
-- USB Peripheral Access Layer  
----- */
```

```
/*!
 * @addtogroup USB_Peripheral_Access_Layer USB Peripheral Access Layer
 * @{
 */

```

```
/** USB - Register Layout Typedef */
```

```
typedef struct {
```

```
__I uint8_t PERID;           /**< Peripheral ID register, offset: 0x0 */

    uint8_t RESERVED_0[3];

__I uint8_t IDCOMP;          /**< Peripheral ID Complement register, offset: 0x4 */

    uint8_t RESERVED_1[3];

__I uint8_t REV;             /**< Peripheral Revision register, offset: 0x8 */

    uint8_t RESERVED_2[3];

__I uint8_t ADDINFO;         /**< Peripheral Additional Info register, offset: 0xC */

    uint8_t RESERVED_3[3];

__IO uint8_t OTGISTAT;       /**< OTG Interrupt Status register, offset: 0x10 */

    uint8_t RESERVED_4[3];

__IO uint8_t OTGICR;          /**< OTG Interrupt Control Register, offset: 0x14 */

    uint8_t RESERVED_5[3];

__IO uint8_t OTGSTAT;         /**< OTG Status register, offset: 0x18 */

    uint8_t RESERVED_6[3];

__IO uint8_t OTGCTL;          /**< OTG Control register, offset: 0x1C */

    uint8_t RESERVED_7[99];

__IO uint8_t ISTAT;           /**< Interrupt Status register, offset: 0x80 */

    uint8_t RESERVED_8[3];

__IO uint8_t INTEN;           /**< Interrupt Enable register, offset: 0x84 */

    uint8_t RESERVED_9[3];

__IO uint8_t ERRSTAT;         /**< Error Interrupt Status register, offset: 0x88 */

    uint8_t RESERVED_10[3];

__IO uint8_t ERREN;            /**< Error Interrupt Enable register, offset: 0x8C */

    uint8_t RESERVED_11[3];

__I uint8_t STAT;              /**< Status register, offset: 0x90 */

    uint8_t RESERVED_12[3];

__IO uint8_t CTL;               /**< Control register, offset: 0x94 */

    uint8_t RESERVED_13[3];

__IO uint8_t ADDR;              /**< Address register, offset: 0x98 */
```

```

    uint8_t RESERVED_14[3];

    __IO uint8_t BDTPAGE1;           /**< BDT Page Register 1, offset: 0x9C */

    uint8_t RESERVED_15[3];

    __IO uint8_t FRMNUML;          /**< Frame Number Register Low, offset: 0xA0 */

    uint8_t RESERVED_16[3];

    __IO uint8_t FRMNUMH;          /**< Frame Number Register High, offset: 0xA4 */

    uint8_t RESERVED_17[3];

    __IO uint8_t TOKEN;            /**< Token register, offset: 0xA8 */

    uint8_t RESERVED_18[3];

    __IO uint8_t SOFTHLD;          /**< SOF Threshold Register, offset: 0xAC */

    uint8_t RESERVED_19[3];

    __IO uint8_t BDTPAGE2;          /**< BDT Page Register 2, offset: 0xB0 */

    uint8_t RESERVED_20[3];

    __IO uint8_t BDTPAGE3;          /**< BDT Page Register 3, offset: 0xB4 */

    uint8_t RESERVED_21[11];

struct {                           /* offset: 0xC0, array step: 0x4 */

    __IO uint8_t ENDPT;           /**< Endpoint Control register, array offset: 0xC0, array step:
                                  0x4 */

    uint8_t RESERVED_0[3];

} ENDPOINT[16];

    __IO uint8_t USBCTRL;          /**< USB Control register, offset: 0x100 */

    uint8_t RESERVED_22[3];

    __I uint8_t OBSERVE;          /**< USB OTG Observe register, offset: 0x104 */

    uint8_t RESERVED_23[3];

    __IO uint8_t CONTROL;          /**< USB OTG Control register, offset: 0x108 */

    uint8_t RESERVED_24[3];

    __IO uint8_t USBTRCO;          /**< USB Transceiver Control Register 0, offset: 0x10C */

    uint8_t RESERVED_25[7];

    __IO uint8_t USBFRMADJUST;     /**< Frame Adjust Register, offset: 0x114 */

```

```

} USB_Type, *USB_MemMapPtr;

/*
-----  

-- USB - Register accessor macros  

----- */
  

/*!  

* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros  

* @{
* /  

  

/* USB - Register accessors */  

#define USB_PERID_REG(base) ((base)->PERID)  

#define USB_IDCOMP_REG(base) ((base)->IDCOMP)  

#define USB_REV_REG(base) ((base)->REV)  

#define USB_ADDINFO_REG(base) ((base)->ADDINFO)  

#define USB_OTGISTAT_REG(base) ((base)->OTGISTAT)  

#define USB_OTGICR_REG(base) ((base)->OTGICR)  

#define USB_OTGSTAT_REG(base) ((base)->OTGSTAT)  

#define USB_OTGCTL_REG(base) ((base)->OTGCTL)  

#define USB_ISTAT_REG(base) ((base)->ISTAT)  

#define USB_INTEN_REG(base) ((base)->INTEN)  

#define USB_ERRSTAT_REG(base) ((base)->ERRSTAT)  

#define USB_ERREN_REG(base) ((base)->ERREN)  

#define USB_STAT_REG(base) ((base)->STAT)  

#define USB_CTL_REG(base) ((base)->CTL)  

#define USB_ADDR_REG(base) ((base)->ADDR)  

#define USB_BDTPAGE1_REG(base) ((base)->BDTPAGE1)

```

```
#define USB_FRMNUML_REG(base) ((base)->FRMNUML)
#define USB_FRMNUMH_REG(base) ((base)->FRMNUMH)
#define USB_TOKEN_REG(base) ((base)->TOKEN)
#define USB_SOFTHLD_REG(base) ((base)->SOFTHLD)
#define USB_BDTPAGE2_REG(base) ((base)->BDTPAGE2)
#define USB_BDTPAGE3_REG(base) ((base)->BDTPAGE3)
#define USB_ENDPT_REG(base,index) ((base)->ENDPOINT[index].ENDPT)
#define USB_ENDPT_COUNT 16
#define USB_USBCTRL_REG(base) ((base)->USBCTRL)
#define USB_OBSERVE_REG(base) ((base)->OBSERVE)
#define USB_CONTROL_REG(base) ((base)->CONTROL)
#define USB_USBTRCO_REG(base) ((base)->USBTRCO)
#define USB_USBFRMADJUST_REG(base) ((base)->USBFRMADJUST)
```

```
/*
 * @}
 */
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*
-----  
-- USB Register Masks  
----- */
```

```
/*
 * @addtogroup USB_Register_Masks USB Register Masks
 * @{
 */

```

```
/* PERID Bit Fields */
```

```

#define USB_PERID_ID_MASK          0x3Fu
#define USB_PERID_ID_SHIFT         0
#define USB_PERID_ID_WIDTH         6
#define USB_PERID_ID(x) (((uint8_t)((uint8_t)(x))<<USB_PERID_ID_SHIFT))&USB_PERID_ID_MASK

/* IDCOMP Bit Fields */

#define USB_IDCOMP_NID_MASK        0x3Fu
#define USB_IDCOMP_NID_SHIFT       0
#define USB_IDCOMP_NID_WIDTH       6
#define USB_IDCOMP_NID(x) (((uint8_t)((uint8_t)(x))<<USB_IDCOMP_NID_SHIFT))&USB_IDCOMP_NID_MASK

/* REV Bit Fields */

#define USB_REV_REV_MASK           0xFFu
#define USB_REV_REV_SHIFT          0
#define USB_REV_REV_WIDTH          8
#define USB_REV_REV(x) (((uint8_t)((uint8_t)(x))<<USB_REV_REV_SHIFT))&USB_REV_REV_MASK

/* ADDINFO Bit Fields */

#define USB_ADDINFO_IHOST_MASK      0x1u
#define USB_ADDINFO_IHOST_SHIFT     0
#define USB_ADDINFO_IHOST_WIDTH     1
#define USB_ADDINFO_IHOST(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IHOST_SHIFT))&USB_ADDINFO_IHOST_MASK

#define USB_ADDINFO_IRQNUM_MASK     0xF8u
#define USB_ADDINFO_IRQNUM_SHIFT    3
#define USB_ADDINFO_IRQNUM_WIDTH    5
#define USB_ADDINFO_IRQNUM(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IRQNUM_SHIFT))&USB_ADDINFO_IRQNUM_MASK

/* OTGISTAT Bit Fields */

#define USB_OTGISTAT_AVBUSCHG_MASK  0x1u

```

```

#define USB_OTGISTAT_AVBUSCHG_SHIFT      0
#define USB_OTGISTAT_AVBUSCHG_WIDTH      1
#define USB_OTGISTAT_AVBUSCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_AVBUSCHG_SHIFT))&USB_OTGISTAT_AVBUSCHG_MASK)

#define USB_OTGISTAT_B_SESS_CHG_MASK      0x4u
#define USB_OTGISTAT_B_SESS_CHG_SHIFT     2
#define USB_OTGISTAT_B_SESS_CHG_WIDTH    1
#define USB_OTGISTAT_B_SESS_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_B_SESS_CHG_SHIFT))&USB_OTGISTAT_B_SESS_CHG_MASK)

#define USB_OTGISTAT_SESSVLDCHG_MASK      0x8u
#define USB_OTGISTAT_SESSVLDCHG_SHIFT     3
#define USB_OTGISTAT_SESSVLDCHG_WIDTH    1
#define USB_OTGISTAT_SESSVLDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_SESSVLDCHG_SHIFT))&USB_OTGISTAT_SESSVLDCHG_MASK)

#define USB_OTGISTAT_LINE_STATE_CHG_MASK   0x20u
#define USB_OTGISTAT_LINE_STATE_CHG_SHIFT  5
#define USB_OTGISTAT_LINE_STATE_CHG_WIDTH  1
#define USB_OTGISTAT_LINE_STATE_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_LINE_STATE_CHG_SHIFT))&USB_OTGISTAT_LINE_STATE_CHG_MASK)

#define USB_OTGISTAT_ONEMSEC_MASK         0x40u
#define USB_OTGISTAT_ONEMSEC_SHIFT       6
#define USB_OTGISTAT_ONEMSEC_WIDTH      1
#define USB_OTGISTAT_ONEMSEC(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_ONEMSEC_SHIFT))&USB_OTGISTAT_ONEMSEC_MASK)

#define USB_OTGISTAT_IDCHG_MASK          0x80u
#define USB_OTGISTAT_IDCHG_SHIFT        7
#define USB_OTGISTAT_IDCHG_WIDTH       1
#define USB_OTGISTAT_IDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_IDCHG_SHIFT))&USB_OTGISTAT_IDCHG_MASK)

/* OTGICR Bit Fields */

```

```

#define USB_OTGICR_AVBUSEN_MASK      0x1u
#define USB_OTGICR_AVBUSEN_SHIFT     0
#define USB_OTGICR_AVBUSEN_WIDTH     1
#define USB_OTGICR_AVBUSEN(x)        (((uint8_t)((uint8_t)(x))<<USB_OTGICR_AVBUSEN_SHIFT))&USB_OTGICR_AVBUSEN_MASK)
#define USB_OTGICR_BSESSEN_MASK      0x4u
#define USB_OTGICR_BSESSEN_SHIFT     2
#define USB_OTGICR_BSESSEN_WIDTH     1
#define USB_OTGICR_BSESSEN(x)        (((uint8_t)((uint8_t)(x))<<USB_OTGICR_BSESSEN_SHIFT))&USB_OTGICR_BSESSEN_MASK)
#define USB_OTGICR_SESSVLDEN_MASK    0x8u
#define USB_OTGICR_SESSVLDEN_SHIFT   3
#define USB_OTGICR_SESSVLDEN_WIDTH   1
#define USB_OTGICR_SESSVLDEN(x)      (((uint8_t)((uint8_t)(x))<<USB_OTGICR_SESSVLDEN_SHIFT))&USB_OTGICR_SESSVLDEN_MASK)
#define USB_OTGICR_LINESTATEEN_MASK   0x20u
#define USB_OTGICR_LINESTATEEN_SHIFT  5
#define USB_OTGICR_LINESTATEEN_WIDTH  1
#define USB_OTGICR_LINESTATEEN(x)     (((uint8_t)((uint8_t)(x))<<USB_OTGICR_LINESTATEEN_SHIFT))&USB_OTGICR_LINESTATEEN_MASK)
#define USB_OTGICR_ONEMSECEN_MASK    0x40u
#define USB_OTGICR_ONEMSECEN_SHIFT   6
#define USB_OTGICR_ONEMSECEN_WIDTH   1
#define USB_OTGICR_ONEMSECEN(x)      (((uint8_t)((uint8_t)(x))<<USB_OTGICR_ONEMSECEN_SHIFT))&USB_OTGICR_ONEMSECEN_MASK)
#define USB_OTGICR_IDEN_MASK         0x80u
#define USB_OTGICR_IDEN_SHIFT        7
#define USB_OTGICR_IDEN_WIDTH        1
#define USB_OTGICR_IDEN(x)          (((uint8_t)((uint8_t)(x))<<USB_OTGICR_IDEN_SHIFT))&USB_OTGICR_IDEN_MASK)
/* OTGSTAT Bit Fields */

```

```

#define USB_OTGSTAT_AVBUSVLD_MASK      0x1u
#define USB_OTGSTAT_AVBUSVLD_SHIFT     0
#define USB_OTGSTAT_AVBUSVLD_WIDTH     1
#define USB_OTGSTAT_AVBUSVLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_AVBUSVLD_SHIFT))&USB_OTGSTAT_AVBUSVLD_MASK

#define USB_OTGSTAT_BSESSEND_MASK       0x4u
#define USB_OTGSTAT_BSESSEND_SHIFT     2
#define USB_OTGSTAT_BSESSEND_WIDTH     1
#define USB_OTGSTAT_BSESSEND(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_BSESSEND_SHIFT))&USB_OTGSTAT_BSESSEND_MASK

#define USB_OTGSTAT_SESS_VLD_MASK       0x8u
#define USB_OTGSTAT_SESS_VLD_SHIFT     3
#define USB_OTGSTAT_SESS_VLD_WIDTH     1
#define USB_OTGSTAT_SESS_VLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_SESS_VLD_SHIFT))&USB_OTGSTAT_SESS_VLD_MASK

#define USB_OTGSTAT_LINESTATESTABLE_MASK 0x20u
#define USB_OTGSTAT_LINESTATESTABLE_SHIFT 5
#define USB_OTGSTAT_LINESTATESTABLE_WIDTH 1
#define USB_OTGSTAT_LINESTATESTABLE(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_LINESTATESTABLE_SHIFT))&USB_OTGSTAT_LINESTATESTABLE_MASK

#define USB_OTGSTAT_ONEMSECEN_MASK      0x40u
#define USB_OTGSTAT_ONEMSECEN_SHIFT     6
#define USB_OTGSTAT_ONEMSECEN_WIDTH     1
#define USB_OTGSTAT_ONEMSECEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ONEMSECEN_SHIFT))&USB_OTGSTAT_ONEMSECEN_MASK

#define USB_OTGSTAT_ID_MASK            0x80u
#define USB_OTGSTAT_ID_SHIFT          7
#define USB_OTGSTAT_ID_WIDTH          1
#define USB_OTGSTAT_ID(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ID_SHIFT))&USB_OTGSTAT_ID_MASK

```

```

/* OTGCTL Bit Fields */

#define USB_OTGCTL_OTGEN_MASK          0x4u
#define USB_OTGCTL_OTGEN_SHIFT         2
#define USB_OTGCTL_OTGEN_WIDTH         1
#define USB_OTGCTL_OTGEN(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_OTGEN_SHIFT))&USB_OTGCTL_OTGEN_MASK)

#define USB_OTGCTL_DMLOW_MASK          0x10u
#define USB_OTGCTL_DMLOW_SHIFT         4
#define USB_OTGCTL_DMLOW_WIDTH         1
#define USB_OTGCTL_DMLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DMLOW_SHIFT))&USB_OTGCTL_DMLOW_MASK)

#define USB_OTGCTL_DPLOW_MASK          0x20u
#define USB_OTGCTL_DPLOW_SHIFT         5
#define USB_OTGCTL_DPLOW_WIDTH         1
#define USB_OTGCTL_DPLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPLOW_SHIFT))&USB_OTGCTL_DPLOW_MASK)

#define USB_OTGCTL_DPHIGH_MASK         0x80u
#define USB_OTGCTL_DPHIGH_SHIFT        7
#define USB_OTGCTL_DPHIGH_WIDTH        1
#define USB_OTGCTL_DPHIGH(x)           (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPHIGH_SHIFT))&USB_OTGCTL_DPHIGH_MASK)

/* ISTAT Bit Fields */

#define USB_ISTAT_USBRST_MASK          0x1u
#define USB_ISTAT_USBRST_SHIFT         0
#define USB_ISTAT_USBRST_WIDTH         1
#define USB_ISTAT_USBRST(x)            (((uint8_t)((uint8_t)(x)<<USB_ISTAT_USBRST_SHIFT))&USB_ISTAT_USBRST_MASK)

#define USB_ISTAT_ERROR_MASK           0x2u
#define USB_ISTAT_ERROR_SHIFT          1
#define USB_ISTAT_ERROR_WIDTH          1

```

```

#define USB_ISTAT_ERROR(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ERROR_SHIFT))&USB_ISTAT_ERROR_MASK)

#define USB_ISTAT_SOFTOK_MASK          0x4u
#define USB_ISTAT_SOFTOK_SHIFT        2
#define USB_ISTAT_SOFTOK_WIDTH        1

#define USB_ISTAT_SOFTOK(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SOFTOK_SHIFT))&USB_ISTAT_SOFTOK_MASK)

#define USB_ISTAT_TOKDNE_MASK         0x8u
#define USB_ISTAT_TOKDNE_SHIFT        3
#define USB_ISTAT_TOKDNE_WIDTH        1

#define USB_ISTAT_TOKDNE(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_TOKDNE_SHIFT))&USB_ISTAT_TOKDNE_MASK)

#define USB_ISTAT_SLEEP_MASK          0x10u
#define USB_ISTAT_SLEEP_SHIFT         4
#define USB_ISTAT_SLEEP_WIDTH         1

#define USB_ISTAT_SLEEP(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SLEEP_SHIFT))&USB_ISTAT_SLEEP_MASK)

#define USB_ISTAT_RESUME_MASK         0x20u
#define USB_ISTAT_RESUME_SHIFT        5
#define USB_ISTAT_RESUME_WIDTH        1

#define USB_ISTAT_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_RESUME_SHIFT))&USB_ISTAT_RESUME_MASK)

#define USB_ISTAT_ATTACH_MASK          0x40u
#define USB_ISTAT_ATTACH_SHIFT         6
#define USB_ISTAT_ATTACH_WIDTH         1

#define USB_ISTAT_ATTACH(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ATTACH_SHIFT))&USB_ISTAT_ATTACH_MASK)

#define USB_ISTAT_STALL_MASK          0x80u
#define USB_ISTAT_STALL_SHIFT         7
#define USB_ISTAT_STALL_WIDTH         1

```

```

#define USB_ISTAT_STALL(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_STALL_SHIFT))&USB_ISTAT_STALL_MASK)

/* INTEN Bit Fields */

#define USB_INTEN_USBRSTEN_MASK          0x1u
#define USB_INTEN_USBRSTEN_SHIFT         0
#define USB_INTEN_USBRSTEN_WIDTH         1
#define USB_INTEN_USBRSTEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_USBRSTEN_SHIFT))&USB_INTEN_USBRSTEN_MASK)

#define USB_INTEN_ERROREN_MASK           0x2u
#define USB_INTEN_ERROREN_SHIFT          1
#define USB_INTEN_ERROREN_WIDTH          1
#define USB_INTEN_ERROREN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ERROREN_SHIFT))&USB_INTEN_ERROREN_MASK)

#define USB_INTEN_SOFTOKEN_MASK          0x4u
#define USB_INTEN_SOFTOKEN_SHIFT         2
#define USB_INTEN_SOFTOKEN_WIDTH         1
#define USB_INTEN_SOFTOKEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SOFTOKEN_SHIFT))&USB_INTEN_SOFTOKEN_MASK)

#define USB_INTEN_TOKDNEEN_MASK          0x8u
#define USB_INTEN_TOKDNEEN_SHIFT         3
#define USB_INTEN_TOKDNEEN_WIDTH         1
#define USB_INTEN_TOKDNEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_TOKDNEEN_SHIFT))&USB_INTEN_TOKDNEEN_MASK)

#define USB_INTEN_SLEEPEN_MASK           0x10u
#define USB_INTEN_SLEEPEN_SHIFT          4
#define USB_INTEN_SLEEPEN_WIDTH          1
#define USB_INTEN_SLEEPEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SLEEPEN_SHIFT))&USB_INTEN_SLEEPEN_MASK)

#define USB_INTEN_RESUMEEN_MASK          0x20u
#define USB_INTEN_RESUMEEN_SHIFT         5
#define USB_INTEN_RESUMEEN_WIDTH         1

```

```

#define USB_INTEN_RESUMEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_RESUMEEN_SHIFT))&USB_INTEN_RESUMEEN_MASK)

#define USB_INTEN_ATTACHEN_MASK          0x40u
#define USB_INTEN_ATTACHEN_SHIFT        6
#define USB_INTEN_ATTACHEN_WIDTH        1

#define USB_INTEN_ATTACHEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ATTACHEN_SHIFT))&USB_INTEN_ATTACHEN_MASK)

#define USB_INTEN_STALLEN_MASK          0x80u
#define USB_INTEN_STALLEN_SHIFT        7
#define USB_INTEN_STALLEN_WIDTH        1

#define USB_INTEN_STALLEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_STALLEN_SHIFT))&USB_INTEN_STALLEN_MASK)

/* ERRSTAT Bit Fields */

#define USB_ERRSTAT_PIDERR_MASK         0x1u
#define USB_ERRSTAT_PIDERR_SHIFT       0
#define USB_ERRSTAT_PIDERR_WIDTH       1

#define USB_ERRSTAT_PIDERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_PIDERR_SHIFT))&USB_ERRSTAT_PIDERR_MASK)

#define USB_ERRSTAT_CRC5EOF_MASK         0x2u
#define USB_ERRSTAT_CRC5EOF_SHIFT       1
#define USB_ERRSTAT_CRC5EOF_WIDTH       1

#define USB_ERRSTAT_CRC5EOF(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC5EOF_SHIFT))&USB_ERRSTAT_CRC5EOF_MASK)

#define USB_ERRSTAT_CRC16_MASK          0x4u
#define USB_ERRSTAT_CRC16_SHIFT        2
#define USB_ERRSTAT_CRC16_WIDTH        1

#define USB_ERRSTAT_CRC16(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC16_SHIFT))&USB_ERRSTAT_CRC16_MASK)

#define USB_ERRSTAT_DFN8_MASK           0x8u
#define USB_ERRSTAT_DFN8_SHIFT         3
#define USB_ERRSTAT_DFN8_WIDTH         1

```

```

#define USB_ERRSTAT_DFN8(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DFN8_SHIFT))&USB_ERRSTAT_DFN8_MASK)

#define USB_ERRSTAT_BTOERR_MASK          0x10u
#define USB_ERRSTAT_BTOERR_SHIFT        4
#define USB_ERRSTAT_BTOERR_WIDTH        1

#define USB_ERRSTAT_BTOERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTOERR_SHIFT))&USB_ERRSTAT_BTOERR_MASK)

#define USB_ERRSTAT_DMAERR_MASK         0x20u
#define USB_ERRSTAT_DMAERR_SHIFT       5
#define USB_ERRSTAT_DMAERR_WIDTH       1

#define USB_ERRSTAT_DMAERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DMAERR_SHIFT))&USB_ERRSTAT_DMAERR_MASK)

#define USB_ERRSTAT_BTSERR_MASK         0x80u
#define USB_ERRSTAT_BTSERR_SHIFT       7
#define USB_ERRSTAT_BTSERR_WIDTH       1

#define USB_ERRSTAT_BTSERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTSERR_SHIFT))&USB_ERRSTAT_BTSERR_MASK)

/* ERREN Bit Fields */

#define USB_ERREN_PIDERREN_MASK         0x1u
#define USB_ERREN_PIDERREN_SHIFT        0
#define USB_ERREN_PIDERREN_WIDTH        1

#define USB_ERREN_PIDERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_PIDERREN_SHIFT))&USB_ERREN_PIDERREN_MASK)

#define USB_ERREN_CRC5EOFEN_MASK        0x2u
#define USB_ERREN_CRC5EOFEN_SHIFT      1
#define USB_ERREN_CRC5EOFEN_WIDTH      1

#define USB_ERREN_CRC5EOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_CRC5EOFEN_SHIFT))&USB_ERREN_CRC5EOFEN_MASK)

#define USB_ERREN_CRC16EN_MASK          0x4u
#define USB_ERREN_CRC16EN_SHIFT        2
#define USB_ERREN_CRC16EN_WIDTH        1

```

```

#define USB_ERREN_CRC16EN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_CRC16EN_SHIFT))&USB_ERREN_CRC16EN_MASK)

#define USB_ERREN_DFN8EN_MASK          0x8u
#define USB_ERREN_DFN8EN_SHIFT         3
#define USB_ERREN_DFN8EN_WIDTH         1

#define USB_ERREN_DFN8EN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_DFN8EN_SHIFT))&USB_ERREN_DFN8EN_MASK)

#define USB_ERREN_BTOERREN_MASK        0x10u
#define USB_ERREN_BTOERREN_SHIFT       4
#define USB_ERREN_BTOERREN_WIDTH       1

#define USB_ERREN_BTOERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_BTOERREN_SHIFT))&USB_ERREN_BTOERREN_MASK)

#define USB_ERREN_DMAERREN_MASK        0x20u
#define USB_ERREN_DMAERREN_SHIFT       5
#define USB_ERREN_DMAERREN_WIDTH       1

#define USB_ERREN_DMAERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_DMAERREN_SHIFT))&USB_ERREN_DMAERREN_MASK)

#define USB_ERREN_BTSERREN_MASK        0x80u
#define USB_ERREN_BTSERREN_SHIFT       7
#define USB_ERREN_BTSERREN_WIDTH       1

#define USB_ERREN_BTSERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_BTSERREN_SHIFT))&USB_ERREN_BTSERREN_MASK)

/* STAT Bit Fields */

#define USB_STAT_ODD_MASK             0x4u
#define USB_STAT_ODD_SHIFT             2
#define USB_STAT_ODD_WIDTH             1

#define USB_STAT_ODD(x)
(((uint8_t)((uint8_t)(x))<<USB_STAT_ODD_SHIFT))&USB_STAT_ODD_MASK)

#define USB_STAT_TX_MASK              0x8u
#define USB_STAT_TX_SHIFT              3
#define USB_STAT_TX_WIDTH              1

```

```

#define USB_STAT_TX(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_TX_SHIFT))&USB_STAT_TX_MASK)

#define USB_STAT_ENDP_MASK          0xF0u
#define USB_STAT_ENDP_SHIFT        4
#define USB_STAT_ENDP_WIDTH        4

#define USB_STAT_ENDP(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_ENDP_SHIFT))&USB_STAT_ENDP_MASK)

/* CTL Bit Fields */

#define USB_CTL_USBENSOFEN_MASK      0x1u
#define USB_CTL_USBENSOFEN_SHIFT    0
#define USB_CTL_USBENSOFEN_WIDTH    1

#define USB_CTL_USBENSOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_USBENSOFEN_SHIFT))&USB_CTL_USBENSOFEN_MASK)

#define USB_CTL_ODDRST_MASK         0x2u
#define USB_CTL_ODDRST_SHIFT       1
#define USB_CTL_ODDRST_WIDTH       1

#define USB_CTL_ODDRST(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_ODDRST_SHIFT))&USB_CTL_ODDRST_MASK)

#define USB_CTL_RESUME_MASK         0x4u
#define USB_CTL_RESUME_SHIFT       2
#define USB_CTL_RESUME_WIDTH       1

#define USB_CTL_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESUME_SHIFT))&USB_CTL_RESUME_MASK)

#define USB_CTL_HOSTMODEEN_MASK     0x8u
#define USB_CTL_HOSTMODEEN_SHIFT   3
#define USB_CTL_HOSTMODEEN_WIDTH   1

#define USB_CTL_HOSTMODEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_HOSTMODEEN_SHIFT))&USB_CTL_HOSTMODEEN_MASK)

#define USB_CTL_RESET_MASK          0x10u
#define USB_CTL_RESET_SHIFT        4
#define USB_CTL_RESET_WIDTH        1

```

```

#define USB_CTL_RESET(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESET_SHIFT))&USB_CTL_RESET_MASK)

#define USB_CTL_TXSUSPENDTOKENBUSY_MASK      0x20u
#define USB_CTL_TXSUSPENDTOKENBUSY_SHIFT     5
#define USB_CTL_TXSUSPENDTOKENBUSY_WIDTH     1

#define USB_CTL_TXSUSPENDTOKENBUSY(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_TXSUSPENDTOKENBUSY_SHIFT))&USB_CTL_TXSUSPENDTOKENBUSY
 _MASK)

#define USB_CTL_SE0_MASK                    0x40u
#define USB_CTL_SE0_SHIFT                  6
#define USB_CTL_SE0_WIDTH                 1

#define USB_CTL_SE0(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_SE0_SHIFT))&USB_CTL_SE0_MASK)

#define USB_CTL_JSTATE_MASK                0x80u
#define USB_CTL_JSTATE_SHIFT              7
#define USB_CTL_JSTATE_WIDTH              1

#define USB_CTL_JSTATE(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_JSTATE_SHIFT))&USB_CTL_JSTATE_MASK)

/* ADDR Bit Fields */

#define USB_ADDR_ADDR_MASK                0x7Fu
#define USB_ADDR_ADDR_SHIFT              0
#define USB_ADDR_ADDR_WIDTH              7

#define USB_ADDR_ADDR(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_ADDR_SHIFT))&USB_ADDR_ADDR_MASK)

#define USB_ADDR_LSEN_MASK                0x80u
#define USB_ADDR_LSEN_SHIFT              7
#define USB_ADDR_LSEN_WIDTH              1

#define USB_ADDR_LSEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_LSEN_SHIFT))&USB_ADDR_LSEN_MASK)

/* BDTPAGE1 Bit Fields */

#define USB_BDTPAGE1_BDTBA_MASK          0xFEu

```

```

#define USB_BDTPAGE1_BDTBA_SHIFT      1
#define USB_BDTPAGE1_BDTBA_WIDTH      7
#define USB_BDTPAGE1_BDTBA(x)
(((uint8_t)((uint8_t)(x)<<USB_BDTPAGE1_BDTBA_SHIFT))&USB_BDTPAGE1_BDTBA_MASK)

/* FRMNUML Bit Fields */

#define USB_FRMNUML_FRM_MASK          0xFFu
#define USB_FRMNUML_FRM_SHIFT         0
#define USB_FRMNUML_FRM_WIDTH         8
#define USB_FRMNUML_FRM(x)
(((uint8_t)((uint8_t)(x)<<USB_FRMNUML_FRM_SHIFT))&USB_FRMNUML_FRM_MASK)

/* FRMNUMH Bit Fields */

#define USB_FRMNUMH_FRM_MASK          0x7u
#define USB_FRMNUMH_FRM_SHIFT         0
#define USB_FRMNUMH_FRM_WIDTH         3
#define USB_FRMNUMH_FRM(x)
(((uint8_t)((uint8_t)(x)<<USB_FRMNUMH_FRM_SHIFT))&USB_FRMNUMH_FRM_MASK)

/* TOKEN Bit Fields */

#define USB_TOKEN_TOKENENDPT_MASK      0xFu
#define USB_TOKEN_TOKENENDPT_SHIFT     0
#define USB_TOKEN_TOKENENDPT_WIDTH     4
#define USB_TOKEN_TOKENENDPT(x)
(((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENENDPT_SHIFT))&USB_TOKEN_TOKENENDPT_MASK)

#define USB_TOKEN_TOKENPID_MASK         0xF0u
#define USB_TOKEN_TOKENPID_SHIFT        4
#define USB_TOKEN_TOKENPID_WIDTH        4
#define USB_TOKEN_TOKENPID(x)
(((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENPID_SHIFT))&USB_TOKEN_TOKENPID_MASK)

/* SOFTHLD Bit Fields */

#define USB_SOFTHLD_CNT_MASK           0xFFu
#define USB_SOFTHLD_CNT_SHIFT          0

```

```

#define USB_SOFTHLD_CNT_WIDTH          8
#define USB_SOFTHLD_CNT(x) (((uint8_t)((uint8_t)(x)<<USB_SOFTHLD_CNT_SHIFT))&USB_SOFTHLD_CNT_MASK)

/* BDTPAGE2 Bit Fields */

#define USB_BDTPAGE2_BDTBA_MASK        0xFFu
#define USB_BDTPAGE2_BDTBA_SHIFT       0
#define USB_BDTPAGE2_BDTBA_WIDTH       8
#define USB_BDTPAGE2_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE2_BDTBA_SHIFT))&USB_BDTPAGE2_BDTBA_MASK)

/* BDTPAGE3 Bit Fields */

#define USB_BDTPAGE3_BDTBA_MASK        0xFFu
#define USB_BDTPAGE3_BDTBA_SHIFT       0
#define USB_BDTPAGE3_BDTBA_WIDTH       8
#define USB_BDTPAGE3_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE3_BDTBA_SHIFT))&USB_BDTPAGE3_BDTBA_MASK)

/* ENDPT Bit Fields */

#define USB_ENDPT_EPHSHK_MASK          0x1u
#define USB_ENDPT_EPHSHK_SHIFT         0
#define USB_ENDPT_EPHSHK_WIDTH         1
#define USB_ENDPT_EPHSHK(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPHSHK_SHIFT))&USB_ENDPT_EPHSHK_MASK)

#define USB_ENDPT_EPSTALL_MASK         0x2u
#define USB_ENDPT_EPSTALL_SHIFT        1
#define USB_ENDPT_EPSTALL_WIDTH        1
#define USB_ENDPT_EPSTALL(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPSTALL_SHIFT))&USB_ENDPT_EPSTALL_MASK)

#define USB_ENDPT_EPTXEN_MASK          0x4u
#define USB_ENDPT_EPTXEN_SHIFT         2
#define USB_ENDPT_EPTXEN_WIDTH         1
#define USB_ENDPT_EPTXEN(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPTXEN_SHIFT))&USB_ENDPT_EPTXEN_MASK)

```

```

#define USB_ENDPT_EPRXEN_MASK          0x8u
#define USB_ENDPT_EPRXEN_SHIFT         3
#define USB_ENDPT_EPRXEN_WIDTH         1
#define USB_ENDPT_EPRXEN(x)            (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPRXEN_SHIFT))&USB_ENDPT_EPRXEN_MASK)

#define USB_ENDPT_EPCTLDIS_MASK        0x10u
#define USB_ENDPT_EPCTLDIS_SHIFT       4
#define USB_ENDPT_EPCTLDIS_WIDTH       1
#define USB_ENDPT_EPCTLDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPCTLDIS_SHIFT))&USB_ENDPT_EPCTLDIS_MASK)

#define USB_ENDPT_RETRYDIS_MASK        0x40u
#define USB_ENDPT_RETRYDIS_SHIFT       6
#define USB_ENDPT_RETRYDIS_WIDTH       1
#define USB_ENDPT_RETRYDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_RETRYDIS_SHIFT))&USB_ENDPT_RETRYDIS_MASK)

#define USB_ENDPT_HOSTWOHUB_MASK        0x80u
#define USB_ENDPT_HOSTWOHUB_SHIFT       7
#define USB_ENDPT_HOSTWOHUB_WIDTH       1
#define USB_ENDPT_HOSTWOHUB(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_HOSTWOHUB_SHIFT))&USB_ENDPT_HOSTWOHUB_MASK)

/* USBCTRL Bit Fields */

#define USB_USBCTRL_PDE_MASK           0x40u
#define USB_USBCTRL_PDE_SHIFT          6
#define USB_USBCTRL_PDE_WIDTH          1
#define USB_USBCTRL_PDE(x)             (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_PDE_SHIFT))&USB_USBCTRL_PDE_MASK)

#define USB_USBCTRL_SUSP_MASK          0x80u
#define USB_USBCTRL_SUSP_SHIFT         7
#define USB_USBCTRL_SUSP_WIDTH         1
#define USB_USBCTRL_SUSP(x)            (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_SUSP_SHIFT))&USB_USBCTRL_SUSP_MASK)

```

```

/* OBSERVE Bit Fields */

#define USB_OBSERVE_DMPD_MASK          0x10u
#define USB_OBSERVE_DMPD_SHIFT         4
#define USB_OBSERVE_DMPD_WIDTH         1
#define USB_OBSERVE_DMPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DMPD_SHIFT))&USB_OBSERVE_DMPD_MASK

#define USB_OBSERVE_DPPD_MASK          0x40u
#define USB_OBSERVE_DPPD_SHIFT         6
#define USB_OBSERVE_DPPD_WIDTH         1
#define USB_OBSERVE_DPPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPD_SHIFT))&USB_OBSERVE_DPPD_MASK

#define USB_OBSERVE_DPPU_MASK          0x80u
#define USB_OBSERVE_DPPU_SHIFT         7
#define USB_OBSERVE_DPPU_WIDTH         1
#define USB_OBSERVE_DPPU(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPU_SHIFT))&USB_OBSERVE_DPPU_MASK

/* CONTROL Bit Fields */

#define USB_CONTROL_DPPULLUPNONOTG_MASK 0x10u
#define USB_CONTROL_DPPULLUPNONOTG_SHIFT 4
#define USB_CONTROL_DPPULLUPNONOTG_WIDTH 1
#define USB_CONTROL_DPPULLUPNONOTG(x)   (((uint8_t)((uint8_t)(x))<<USB_CONTROL_DPPULLUPNONOTG_SHIFT))&USB_CONTROL_DPPULLUPNONOTG_MASK

/* USBTRCO Bit Fields */

#define USB_USBTRCO_USB_RESUME_INT_MASK 0x1u
#define USB_USBTRCO_USB_RESUME_INT_SHIFT 0
#define USB_USBTRCO_USB_RESUME_INT_WIDTH 1
#define USB_USBTRCO_USB_RESUME_INT(x)   (((uint8_t)((uint8_t)(x))<<USB_USBTRCO_USB_RESUME_INT_SHIFT))&USB_USBTRCO_USB_RESUME_INT_MASK

#define USB_USBTRCO_SYNC_DET_MASK      0x2u

```

```

#define USB_USBTRCO_SYNC_DET_SHIFT      1
#define USB_USBTRCO_SYNC_DET_WIDTH      1
#define USB_USBTRCO_SYNC_DET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_SYNC_DET_SHIFT))&USB_USBTRCO_SYNC_DET_MASK)

#define USB_USBTRCO_USBRESMEN_MASK      0x20u
#define USB_USBTRCO_USBRESMEN_SHIFT      5
#define USB_USBTRCO_USBRESMEN_WIDTH      1
#define USB_USBTRCO_USBRESMEN(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESMEN_SHIFT))&USB_USBTRCO_USBRESMEN_MASK)

#define USB_USBTRCO_USBRESET_MASK        0x80u
#define USB_USBTRCO_USBRESET_SHIFT       7
#define USB_USBTRCO_USBRESET_WIDTH       1
#define USB_USBTRCO_USBRESET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESET_SHIFT))&USB_USBTRCO_USBRESET_MASK)

/* USBFRMADJUST Bit Fields */

#define USB_USBFRMADJUST_ADJ_MASK        0xFFu
#define USB_USBFRMADJUST_ADJ_SHIFT       0
#define USB_USBFRMADJUST_ADJ_WIDTH       8
#define USB_USBFRMADJUST_ADJ(x)
(((uint8_t)((uint8_t)(x)<<USB_USBFRMADJUST_ADJ_SHIFT))&USB_USBFRMADJUST_ADJ_MASK)

/*
 * @}
 */

/* end of group USB_Register_Masks */

```

```

/* USB - Peripheral instance base addresses */
/** Peripheral USBO base address */
#define USBO_BASE                      (0x40072000u)
/** Peripheral USBO base pointer */

```

```

#define USBO ((USB_Type *)USBO_BASE)

#define USBO_BASE_PTR (USBO)

/** Array initializer of USB peripheral base addresses */

#define USB_BASE_ADDRS { USBO_BASE }

/** Array initializer of USB peripheral base pointers */

#define USB_BASE_PTRS { USBO }

/*
-----  

-- USB - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros  

* @{
* /
```

```

/* USB - Register instance definitions */

/* USBO */

#define USBO_PERID USB_PERID_REG(USBO)
#define USBO_IDCOMP USB_IDCOMP_REG(USBO)
#define USBO_REV USB_REV_REG(USBO)
#define USBO_ADDINFO USB_ADDINFO_REG(USBO)
#define USBO_OTGISTAT USB_OTGISTAT_REG(USBO)
#define USBO_OTGICR USB_OTGICR_REG(USBO)
#define USBO_OTGSTAT USB_OTGSTAT_REG(USBO)
#define USBO_OTGCTL USB_OTGCTL_REG(USBO)
#define USBO_ISTAT USB_ISTAT_REG(USBO)
#define USBO_INTEN USB_INTEN_REG(USBO)
```

#define USBO_ERRSTAT	USB_ERRSTAT_REG(USBO)
#define USBO_ERREN	USB_ERREN_REG(USBO)
#define USBO_STAT	USB_STAT_REG(USBO)
#define USBO_CTL	USB_CTL_REG(USBO)
#define USBO_ADDR	USB_ADDR_REG(USBO)
#define USBO_BDTPAGE1	USB_BDTPAGE1_REG(USBO)
#define USBO_FRMNUML	USB_FRMNUML_REG(USBO)
#define USBO_FRMNUMH	USB_FRMNUMH_REG(USBO)
#define USBO_TOKEN	USB_TOKEN_REG(USBO)
#define USBO_SOFTHLD	USB_SOFTHLD_REG(USBO)
#define USBO_BDTPAGE2	USB_BDTPAGE2_REG(USBO)
#define USBO_BDTPAGE3	USB_BDTPAGE3_REG(USBO)
#define USBO_ENDPT0	USB_ENDPT_REG(USBO,0)
#define USBO_ENDPT1	USB_ENDPT_REG(USBO,1)
#define USBO_ENDPT2	USB_ENDPT_REG(USBO,2)
#define USBO_ENDPT3	USB_ENDPT_REG(USBO,3)
#define USBO_ENDPT4	USB_ENDPT_REG(USBO,4)
#define USBO_ENDPT5	USB_ENDPT_REG(USBO,5)
#define USBO_ENDPT6	USB_ENDPT_REG(USBO,6)
#define USBO_ENDPT7	USB_ENDPT_REG(USBO,7)
#define USBO_ENDPT8	USB_ENDPT_REG(USBO,8)
#define USBO_ENDPT9	USB_ENDPT_REG(USBO,9)
#define USBO_ENDPT10	USB_ENDPT_REG(USBO,10)
#define USBO_ENDPT11	USB_ENDPT_REG(USBO,11)
#define USBO_ENDPT12	USB_ENDPT_REG(USBO,12)
#define USBO_ENDPT13	USB_ENDPT_REG(USBO,13)
#define USBO_ENDPT14	USB_ENDPT_REG(USBO,14)
#define USBO_ENDPT15	USB_ENDPT_REG(USBO,15)
#define USBO_USBCTRL	USB_USBCTRL_REG(USBO)

```
#define USBO_OBSERVE           USB_OBSERVE_REG(USBO)
#define USBO_CONTROL            USB_CONTROL_REG(USBO)
#define USBO_USBTRC0            USB_USBTRC0_REG(USBO)
#define USBO_USBFRMADJUST       USB_USBFRMADJUST_REG(USBO)
```

```
/* USB - Register array accessors */
```

```
#define USBO_ENDPT(index)      USB_ENDPT_REG(USBO,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Peripheral_Access_Layer */
```

```
/*
```

```
** End of section using anonymous unions
```

```
*/
```

```
#if defined(__ARMCC_VERSION)
    #pragma pop
#elif defined(__CWCC__)
    #pragma pop
#elif defined(__GNUC__)
    /* leave anonymous unions enabled */
#elif defined(__IAR_SYSTEMS_ICC__)
```

```

#pragma language=default

#else
#error Not supported compiler type

#endif

/*!
 * @}
 */

/* /* end of group Peripheral_access_layer */

/*
-----  

-- Backward Compatibility  

----- */

```

```

/*!  

 * @addtogroup Backward_Compatibility_Symbols Backward Compatibility  

 * @{
 * /  


```

#define DMA_REQC_ARR_DMAC_MASK	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC_SHIFT	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC(x)	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_MASK	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_SHIFT	This_symbol_has_been_deprecated
#define DMA_REQC0	This_symbol_has_been_deprecated
#define DMA_REQC1	This_symbol_has_been_deprecated
#define DMA_REQC2	This_symbol_has_been_deprecated
#define DMA_REQC3	This_symbol_has_been_deprecated
#define MCG_S_LOLS_MASK	MCG_S_LOLS0_MASK

#define MCG_S_LOLS_SHIFT	MCG_S_LOLS0_SHIFT
#define SIM_FCFG2_MAXADDR_MASK	SIM_FCFG2_MAXADDR0_MASK
#define SIM_FCFG2_MAXADDR_SHIFT	SIM_FCFG2_MAXADDR0_SHIFT
#define SIM_FCFG2_MAXADDR	SIM_FCFG2_MAXADDR0
#define SPI_C2_SPLPIE_MASK	This_symbol_has_been_deprecated
#define SPI_C2_SPLPIE_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_SHIFT	This_symbol_has_been_deprecated
#define UARTLP_Type	UART0_Type
#define UARTLP_BDH_REG	UART0_BDH_REG
#define UARTLP_BDL_REG	UART0_BDL_REG
#define UARTLP_C1_REG	UART0_C1_REG
#define UARTLP_C2_REG	UART0_C2_REG
#define UARTLP_S1_REG	UART0_S1_REG
#define UARTLP_S2_REG	UART0_S2_REG
#define UARTLP_C3_REG	UART0_C3_REG
#define UARTLP_D_REG	UART0_D_REG
#define UARTLP_MA1_REG	UART0_MA1_REG
#define UARTLP_MA2_REG	UART0_MA2_REG
#define UARTLP_C4_REG	UART0_C4_REG
#define UARTLP_C5_REG	UART0_C5_REG
#define UARTLP_BDH_SBR_MASK	UART0_BDH_SBR_MASK
#define UARTLP_BDH_SBR_SHIFT	UART0_BDH_SBR_SHIFT
#define UARTLP_BDH_SBR(x)	UART0_BDH_SBR(x)
#define UARTLP_BDH_SBNS_MASK	UART0_BDH_SBNS_MASK

#define UARLTP_BDH_SBNS_SHIFT	UART0_BDH_SBNS_SHIFT
#define UARLTP_BDH_RXEDGIE_MASK	UART0_BDH_RXEDGIE_MASK
#define UARLTP_BDH_RXEDGIE_SHIFT	UART0_BDH_RXEDGIE_SHIFT
#define UARLTP_BDH_LBKDIE_MASK	UART0_BDH_LBKDIE_MASK
#define UARLTP_BDH_LBKDIE_SHIFT	UART0_BDH_LBKDIE_SHIFT
#define UARLTP_BDL_SBR_MASK	UART0_BDL_SBR_MASK
#define UARLTP_BDL_SBR_SHIFT	UART0_BDL_SBR_SHIFT
#define UARLTP_BDL_SBR(x)	UART0_BDL_SBR(x)
#define UARLTP_C1_PT_MASK	UART0_C1_PT_MASK
#define UARLTP_C1_PT_SHIFT	UART0_C1_PT_SHIFT
#define UARLTP_C1_PE_MASK	UART0_C1_PE_MASK
#define UARLTP_C1_PE_SHIFT	UART0_C1_PE_SHIFT
#define UARLTP_C1_ILT_MASK	UART0_C1_ILT_MASK
#define UARLTP_C1_ILT_SHIFT	UART0_C1_ILT_SHIFT
#define UARLTP_C1_WAKE_MASK	UART0_C1_WAKE_MASK
#define UARLTP_C1_WAKE_SHIFT	UART0_C1_WAKE_SHIFT
#define UARLTP_C1_M_MASK	UART0_C1_M_MASK
#define UARLTP_C1_M_SHIFT	UART0_C1_M_SHIFT
#define UARLTP_C1_RSRC_MASK	UART0_C1_RSRC_MASK
#define UARLTP_C1_RSRC_SHIFT	UART0_C1_RSRC_SHIFT
#define UARLTP_C1_DOZEEN_MASK	UART0_C1_DOZEEN_MASK
#define UARLTP_C1_DOZEEN_SHIFT	UART0_C1_DOZEEN_SHIFT
#define UARLTP_C1_LOOPS_MASK	UART0_C1_LOOPS_MASK
#define UARLTP_C1_LOOPS_SHIFT	UART0_C1_LOOPS_SHIFT
#define UARLTP_C2_SBK_MASK	UART0_C2_SBK_MASK
#define UARLTP_C2_SBK_SHIFT	UART0_C2_SBK_SHIFT
#define UARLTP_C2_RWU_MASK	UART0_C2_RWU_MASK
#define UARLTP_C2_RWU_SHIFT	UART0_C2_RWU_SHIFT
#define UARLTP_C2_RE_MASK	UART0_C2_RE_MASK

#define UARTLP_C2_RE_SHIFT	UART0_C2_RE_SHIFT
#define UARTLP_C2_TE_MASK	UART0_C2_TE_MASK
#define UARTLP_C2_TE_SHIFT	UART0_C2_TE_SHIFT
#define UARTLP_C2_ILIE_MASK	UART0_C2_ILIE_MASK
#define UARTLP_C2_ILIE_SHIFT	UART0_C2_ILIE_SHIFT
#define UARTLP_C2_RIE_MASK	UART0_C2_RIE_MASK
#define UARTLP_C2_RIE_SHIFT	UART0_C2_RIE_SHIFT
#define UARTLP_C2_TCIE_MASK	UART0_C2_TCIE_MASK
#define UARTLP_C2_TCIE_SHIFT	UART0_C2_TCIE_SHIFT
#define UARTLP_C2_TIE_MASK	UART0_C2_TIE_MASK
#define UARTLP_C2_TIE_SHIFT	UART0_C2_TIE_SHIFT
#define UARTLP_S1_PF_MASK	UART0_S1_PF_MASK
#define UARTLP_S1_PF_SHIFT	UART0_S1_PF_SHIFT
#define UARTLP_S1_FE_MASK	UART0_S1_FE_MASK
#define UARTLP_S1_FE_SHIFT	UART0_S1_FE_SHIFT
#define UARTLP_S1_NF_MASK	UART0_S1_NF_MASK
#define UARTLP_S1_NF_SHIFT	UART0_S1_NF_SHIFT
#define UARTLP_S1_OR_MASK	UART0_S1_OR_MASK
#define UARTLP_S1_OR_SHIFT	UART0_S1_OR_SHIFT
#define UARTLP_S1_IDLE_MASK	UART0_S1_IDLE_MASK
#define UARTLP_S1_IDLE_SHIFT	UART0_S1_IDLE_SHIFT
#define UARTLP_S1_RDRF_MASK	UART0_S1_RDRF_MASK
#define UARTLP_S1_RDRF_SHIFT	UART0_S1_RDRF_SHIFT
#define UARTLP_S1_TC_MASK	UART0_S1_TC_MASK
#define UARTLP_S1_TC_SHIFT	UART0_S1_TC_SHIFT
#define UARTLP_S1_TDRE_MASK	UART0_S1_TDRE_MASK
#define UARTLP_S1_TDRE_SHIFT	UART0_S1_TDRE_SHIFT
#define UARTLP_S2_RAF_MASK	UART0_S2_RAF_MASK
#define UARTLP_S2_RAF_SHIFT	UART0_S2_RAF_SHIFT

#define UARLTP_S2_LBKDE_MASK	UART0_S2_LBKDE_MASK
#define UARLTP_S2_LBKDE_SHIFT	UART0_S2_LBKDE_SHIFT
#define UARLTP_S2_BRK13_MASK	UART0_S2_BRK13_MASK
#define UARLTP_S2_BRK13_SHIFT	UART0_S2_BRK13_SHIFT
#define UARLTP_S2_RWUID_MASK	UART0_S2_RWUID_MASK
#define UARLTP_S2_RWUID_SHIFT	UART0_S2_RWUID_SHIFT
#define UARLTP_S2_RXINV_MASK	UART0_S2_RXINV_MASK
#define UARLTP_S2_RXINV_SHIFT	UART0_S2_RXINV_SHIFT
#define UARLTP_S2_MSBF_MASK	UART0_S2_MSBF_MASK
#define UARLTP_S2_MSBF_SHIFT	UART0_S2_MSBF_SHIFT
#define UARLTP_S2_RXEDGIF_MASK	UART0_S2_RXEDGIF_MASK
#define UARLTP_S2_RXEDGIF_SHIFT	UART0_S2_RXEDGIF_SHIFT
#define UARLTP_S2_LBKDIF_MASK	UART0_S2_LBKDIF_MASK
#define UARLTP_S2_LBKDIF_SHIFT	UART0_S2_LBKDIF_SHIFT
#define UARLTP_C3_PEIE_MASK	UART0_C3_PEIE_MASK
#define UARLTP_C3_PEIE_SHIFT	UART0_C3_PEIE_SHIFT
#define UARLTP_C3_FEIE_MASK	UART0_C3_FEIE_MASK
#define UARLTP_C3_FEIE_SHIFT	UART0_C3_FEIE_SHIFT
#define UARLTP_C3_NEIE_MASK	UART0_C3_NEIE_MASK
#define UARLTP_C3_NEIE_SHIFT	UART0_C3_NEIE_SHIFT
#define UARLTP_C3_ORIE_MASK	UART0_C3_ORIE_MASK
#define UARLTP_C3_ORIE_SHIFT	UART0_C3_ORIE_SHIFT
#define UARLTP_C3_TXINV_MASK	UART0_C3_TXINV_MASK
#define UARLTP_C3_TXINV_SHIFT	UART0_C3_TXINV_SHIFT
#define UARLTP_C3_TXDIR_MASK	UART0_C3_TXDIR_MASK
#define UARLTP_C3_TXDIR_SHIFT	UART0_C3_TXDIR_SHIFT
#define UARLTP_C3_R9T8_MASK	UART0_C3_R9T8_MASK
#define UARLTP_C3_R9T8_SHIFT	UART0_C3_R9T8_SHIFT
#define UARLTP_C3_R8T9_MASK	UART0_C3_R8T9_MASK

#define UARTLP_C3_R8T9_SHIFT	UART0_C3_R8T9_SHIFT
#define UARTLP_DROTO_MASK	UART0_DROTO_MASK
#define UARTLP_DROTO_SHIFT	UART0_DROTO_SHIFT
#define UARTLP_DR1T1_MASK	UART0_DR1T1_MASK
#define UARTLP_DR1T1_SHIFT	UART0_DR1T1_SHIFT
#define UARTLP_DR2T2_MASK	UART0_DR2T2_MASK
#define UARTLP_DR2T2_SHIFT	UART0_DR2T2_SHIFT
#define UARTLP_DR3T3_MASK	UART0_DR3T3_MASK
#define UARTLP_DR3T3_SHIFT	UART0_DR3T3_SHIFT
#define UARTLP_DR4T4_MASK	UART0_DR4T4_MASK
#define UARTLP_DR4T4_SHIFT	UART0_DR4T4_SHIFT
#define UARTLP_DR5T5_MASK	UART0_DR5T5_MASK
#define UARTLP_DR5T5_SHIFT	UART0_DR5T5_SHIFT
#define UARTLP_DR6T6_MASK	UART0_DR6T6_MASK
#define UARTLP_DR6T6_SHIFT	UART0_DR6T6_SHIFT
#define UARTLP_DR7T7_MASK	UART0_DR7T7_MASK
#define UARTLP_DR7T7_SHIFT	UART0_DR7T7_SHIFT
#define UARTLP_MA1_MA_MASK	UART0_MA1_MA_MASK
#define UARTLP_MA1_MA_SHIFT	UART0_MA1_MA_SHIFT
#define UARTLP_MA1_MA(x)	UART0_MA1_MA(x)
#define UARTLP_MA2_MA_MASK	UART0_MA2_MA_MASK
#define UARTLP_MA2_MA_SHIFT	UART0_MA2_MA_SHIFT
#define UARTLP_MA2_MA(x)	UART0_MA2_MA(x)
#define UARTLP_C4_OSR_MASK	UART0_C4_OSR_MASK
#define UARTLP_C4_OSR_SHIFT	UART0_C4_OSR_SHIFT
#define UARTLP_C4_OSR(x)	UART0_C4_OSR(x)
#define UARTLP_C4_M10_MASK	UART0_C4_M10_MASK
#define UARTLP_C4_M10_SHIFT	UART0_C4_M10_SHIFT
#define UARTLP_C4_MAEN2_MASK	UART0_C4_MAEN2_MASK

```
#define UARTLP_C4_MAEN2_SHIFT          UART0_C4_MAEN2_SHIFT
#define UARTLP_C4_MAEN1_MASK            UART0_C4_MAEN1_MASK
#define UARTLP_C4_MAEN1_SHIFT          UART0_C4_MAEN1_SHIFT
#define UARTLP_C5_RESETDISC_MASK        UART0_C5_RESETDISC_MASK
#define UARTLP_C5_RESETDISC_SHIFT      UART0_C5_RESETDISC_SHIFT
#define UARTLP_C5_BOTHEDGE_MASK        UART0_C5_BOTHEDGE_MASK
#define UARTLP_C5_BOTHEDGE_SHIFT      UART0_C5_BOTHEDGE_SHIFT
#define UARTLP_C5_RDMAE_MASK           UART0_C5_RDMAE_MASK
#define UARTLP_C5_RDMAE_SHIFT          UART0_C5_RDMAE_SHIFT
#define UARTLP_C5_TDMAE_MASK           UART0_C5_TDMAE_MASK
#define UARTLP_C5_TDMAE_SHIFT          UART0_C5_TDMAE_SHIFT

#define UARTLP_BASES                  UARTLP_BASES
#define NV_FOPT_EZPORT_DIS_MASK        This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT      This_symbol_has_been_deprecated
#define ADC_BASES                     ADC_BASE_PTRS
#define CMP_BASES                     CMP_BASE_PTRS
#define DAC_BASES                     DAC_BASE_PTRS
#define DMA_BASES                     DMA_BASE_PTRS
#define DMAMUX_BASES                 DMAMUX_BASE_PTRS
#define FPTA_BASE_PTR                 GPIOA_BASE_PTR
#define FPTA_BASE                      GPIOA_BASE
#define FPTA                          GPIOA
#define FPTB_BASE_PTR                 GPIOB_BASE_PTR
#define FPTB_BASE                      GPIOB_BASE
#define FPTB                          GPIOB
#define FPTC_BASE_PTR                 GPIOC_BASE_PTR
#define FPTC_BASE                      GPIOC_BASE
#define FPTC                          GPIOC
#define FPTD_BASE_PTR                 GPIOD_BASE_PTR
```

#define FPTD_BASE	FGPIOD_BASE
#define FPTD	FGPIOD
#define FPTE_BASE_PTR	FGPIOE_BASE_PTR
#define FPTE_BASE	FGPIOE_BASE
#define FPTE	FGPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define FTFA_BASES	FTFA_BASE_PTRS
#define PTA_BASE_PTR	GPIOA_BASE_PTR
#define PTA_BASE	GPIOA_BASE
#define PTA	GPIOA
#define PTB_BASE_PTR	GPIOB_BASE_PTR
#define PTB_BASE	GPIOB_BASE
#define PTB	GPIOB
#define PTC_BASE_PTR	GPIOC_BASE_PTR
#define PTC_BASE	GPIOC_BASE
#define PTC	GPIOC
#define PTD_BASE_PTR	GPIOD_BASE_PTR
#define PTD_BASE	GPIOD_BASE
#define PTD	GPIOD
#define PTE_BASE_PTR	GPIOE_BASE_PTR
#define PTE_BASE	GPIOE_BASE
#define PTE	GPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define I2C_BASES	I2C_BASE_PTRS
#define LLWU_BASES	LLWU_BASE_PTRS
#define LPTMR_BASES	LPTMR_BASE_PTRS
#define MCG_BASES	MCG_BASE_PTRS
#define MCM_BASES	MCM_BASE_PTRS
#define MTB_BASES	MTB_BASE_PTRS

```
#define MTBDWT_BASES          MTBDWT_BASE_PTRS
#define NV_BASES                NV_BASES
#define OSC_BASES               OSC_BASE_PTRS
#define PIT_BASES               PIT_BASE_PTRS
#define PMC_BASES               PMC_BASE_PTRS
#define PORT_BASES              PORT_BASE_PTRS
#define RCM_BASES               RCM_BASE_PTRS
#define ROM_BASES               ROM_BASE_PTRS
#define RTC_BASES               RTC_BASE_PTRS
#define SIM_BASES               SIM_BASE_PTRS
#define SMC_BASES               SMC_BASE_PTRS
#define SPI_BASES               SPI_BASE_PTRS
#define TPM_BASES               TPM_BASE_PTRS
#define TSI_BASES               TSI_BASE_PTRS
#define UART_BASES              UART_BASE_PTRS
#define UART0_BASES             UART0_BASE_PTRS
#define USB_BASES               USB_BASE_PTRS
#define LPTimer_IRQn            LPTMR0_IRQn
#define LPTimer_IRQHandler       LPTMR0_IRQHandler
#define LLW_IRQn                LLWU_IRQn
#define LLW_IRQHandler          LLWU_IRQHandler
```

```
/*
 * @}
 */
/* /* end of group Backward_Compatibility_Symbols */
```

```
#else /* #if !defined(MKL25Z4_H_) */
```

```

/* There is already included the same memory map. Check if it is compatible (has the same major
version) */

#if (MCU_MEM_MAP_VERSION != 0x0200u)

#if (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING))

#warning There are included two not compatible versions of memory maps. Please check possible
differences.

#endif /* (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING)) */

#endif /* (MCU_MEM_MAP_VERSION != 0x0200u) */

#endif /* #if !defined(MKL25Z4_H_) */

/* MKL25Z4.h, eof. */

/***********************/

* @file core_cmInstr.h
* @brief CMSIS Cortex-M Core Instruction Access Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*
***********************/

/* Copyright (c) 2009 - 2014 ARM LIMITED

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMINSTR_H  
#define __CORE_CMINSTR_H
```

```
/* ##### Core Instruction Access ##### */  
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface  
Access to dedicated instructions  
@{  
*/
```

```
#if defined (__CC_ARM ) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */
```

```
#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif
```

```
/** \brief No Operation
```

No Operation does nothing. This instruction can be used for code alignment purposes.

```
 */
#define __NOP           __nop
```

```
/** \brief Wait For Interrupt
```

Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occurs.

```
 */
#define __WFI           __wfi
```

```
/** \brief Wait For Event
```

Wait For Event is a hint instruction that permits the processor to enter a low-power state until one of a number of events occurs.

```
 */
#define __WFE           __wfe
```

```
/** \brief Send Event
```

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

```
*/
```

```
#define __SEV           __sev
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the instruction has been completed.

```
*/
```

```
#define __ISB() do {\  
    __schedule_barrier();\  
    __isb(0xF);\  
    __schedule_barrier();\  
} while (0)
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

```
*/
```

```
#define __DSB() do {\  
    __schedule_barrier();\  
    __dsb(0xF);\  
}
```

```
__schedule_barrier();\}
} while (0)
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before and after the instruction, without ensuring their completion.

```
*/
```

```
#define __DMB() do {\n    __schedule_barrier();\n    __dmb(0xF);\\n\n    __schedule_barrier();\\n\n} while (0)
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
#define __REV          __rev
```

```
/** \brief Reverse byte order (16 bit)
```

This function reverses the byte order in two unsigned short values.

```
\param [in]  value Value to reverse
```

```
\return Reversed value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".rev16_text"))) __STATIC_INLINE __ASM uint32_t __REV16(uint32_t value)
{
    rev16 r0, r0
    bx lr
}
#endif
```

```
/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in] value Value to reverse
\return Reversed value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".revsh_text"))) __STATIC_INLINE __ASM int32_t __REVSH(int32_t value)
{
    revsh r0, r0
    bx lr
}
#endif
```

```
/** \brief Rotate Right in unsigned value (32 bit)
```

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```
\param [in]  value Value to rotate
\param [in]  value Number of Bits to rotate
\return      Rotated value
*/
#define __ROR          __ror

/** \brief Breakpoint
```

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

```
\param [in]  value is ignored by the processor.

If required, a debugger can use it to store additional information about the breakpoint.
*/
#define __BKPT(value)      __breakpoint(value)

/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
```

```

#if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)

#define __RBIT          __rbit

#else

__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t value)
{

    uint32_t result;

    int32_t s = 4 /*sizeof(v)*/ * 8 - 1; // extra shift needed at end

    result = value;           // r will be reversed bits of v; first get LSB of v

    for (value >>= 1; value; value >>= 1)

    {

        result <<= 1;

        result |= value & 1;

        s--;

    }

    result <<= s;           // shift when v's highest bits are zero

    return(result);

}

#endif

```

/** \brief Count leading zeros

This function counts the number of leading zeros of a data value.

```

\param [in] value Value to count the leading zeros

\return      number of leading zeros in value

*/
#define __CLZ          __clz

```

```
#if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
```

```
/** \brief LDR Exclusive (8 bit)
```

This function executes a exclusive LDR instruction for 8 bit value.

```
\param [in]  ptr  Pointer to data  
\return     value of type uint8_t at (*ptr)  
*/  
  
#define __LDREXB(ptr)          ((uint8_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (16 bit)
```

This function executes a exclusive LDR instruction for 16 bit values.

```
\param [in]  ptr  Pointer to data  
\return     value of type uint16_t at (*ptr)  
*/  
  
#define __LDREXBH(ptr)         ((uint16_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (32 bit)
```

This function executes a exclusive LDR instruction for 32 bit values.

```
\param [in]  ptr  Pointer to data
```

```
\return value of type uint32_t at (*ptr)
*/
#define __LDREXW(ptr) ((uint32_t ) __ldrex(ptr))
```

```
/** \brief STR Exclusive (8 bit)
```

This function executes a exclusive STR instruction for 8 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
#define __STREXB(value, ptr) __strex(value, ptr)
```

```
/** \brief STR Exclusive (16 bit)
```

This function executes a exclusive STR instruction for 16 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
#define __STREXBH(value, ptr) __strex(value, ptr)
```

```
/** \brief STR Exclusive (32 bit)
```

This function executes a exclusive STR instruction for 32 bit values.

\param [in] value Value to store

\param [in] ptr Pointer to location

\return 0 Function succeeded

\return 1 Function failed

```
*/
```

```
#define __STREXW(value, ptr)      __strex(value, ptr)
```

```
/** \brief Remove the exclusive lock
```

This function removes the exclusive lock which is created by LDREX.

```
*/
```

```
#define __CLREX      __clrex
```

```
/** \brief Signed Saturate
```

This function saturates a signed value.

\param [in] value Value to be saturated

\param [in] sat Bit position to saturate to (1..32)

\return Saturated value

```
*/
```

```
#define __SSAT      __ssat
```

```
/** \brief Unsigned Saturate
```

This function saturates an unsigned value.

```
\param [in] value Value to be saturated  
\param [in] sat Bit position to saturate to (0..31)  
\return Saturated value  
*/  


```
#define __USAT __usat
```


```

```
/** \brief Rotate Right with Extend (32 bit)
```

This function moves each bit of a bitstring right by one bit.

The carry input is shifted in at the left end of the bitstring.

```
\param [in] value Value to rotate  
\return Rotated value  
*/  


```
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".rrx_text"))) __STATIC_INLINE __ASM uint32_t __RRX(uint32_t value)
{
 rrx r0, r0
 bx lr
}
#endif
```


```

```
/** \brief LDRT Unprivileged (8 bit)
```

This function executes a Unprivileged LDRT instruction for 8 bit value.

```
\param [in] ptr Pointer to data  
\return value of type uint8_t at (*ptr)  
*/  
  
#define __LDRBT(ptr) ((uint8_t) __ldrt(ptr))
```

```
/** \brief LDRT Unprivileged (16 bit)
```

This function executes a Unprivileged LDRT instruction for 16 bit values.

```
\param [in] ptr Pointer to data  
\return value of type uint16_t at (*ptr)  
*/  
  
#define __LDRHT(ptr) ((uint16_t) __ldrt(ptr))
```

```
/** \brief LDRT Unprivileged (32 bit)
```

This function executes a Unprivileged LDRT instruction for 32 bit values.

```
\param [in] ptr Pointer to data  
\return value of type uint32_t at (*ptr)  
*/  
  
#define __LDRT(ptr) ((uint32_t) __ldrt(ptr))
```

```
/** \brief STRT Unprivileged (8 bit)
```

This function executes a Unprivileged STRT instruction for 8 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
*/  


```
#define __STRBT(value, ptr) __strt(value, ptr)
```


```

```
/** \brief STRT Unprivileged (16 bit)
```

This function executes a Unprivileged STRT instruction for 16 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
*/  


```
#define __STRHT(value, ptr) __strt(value, ptr)
```


```

```
/** \brief STRT Unprivileged (32 bit)
```

This function executes a Unprivileged STRT instruction for 32 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
*/
```

```

#define __STRT(value, ptr)          __strt(value, ptr)

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#ifndef __thumb2__
/* Define macros for porting to both thumb1 and thumb2.

 * For thumb1, use low register (r0-r7), specified by constraint "l"
 * Otherwise, use general registers, specified by constraint "r" */

#if defined(__thumb__)
#define __CMSIS_GCC_OUT_REG(r) "=l" (r)
#define __CMSIS_GCC_USE_REG(r) "l" (r)
#else
#define __CMSIS_GCC_OUT_REG(r) "=r" (r)
#define __CMSIS_GCC_USE_REG(r) "r" (r)
#endif

/** \brief No Operation
 */

```

No Operation does nothing. This instruction can be used for code alignment purposes.

```

__attribute__((always_inline)) __STATIC_INLINE void __NOP(void)
{
    __ASM volatile ("nop");
}
```

```
/** \brief Wait For Interrupt

Wait For Interrupt is a hint instruction that suspends execution
until one of a number of events occurs.

*/
__attribute__((always_inline)) __STATIC_INLINE void __WFI(void)
{
    __ASM volatile ("wfi");
}
```

```
/** \brief Wait For Event

Wait For Event is a hint instruction that permits the processor to enter
a low-power state until one of a number of events occurs.

*/
__attribute__((always_inline)) __STATIC_INLINE void __WFE(void)
{
    __ASM volatile ("wfe");
}
```

```
/** \brief Send Event

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

*/
__attribute__((always_inline)) __STATIC_INLINE void __SEV(void)
{
    __ASM volatile ("sev");
```

```
}
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the instruction has been completed.

```
*/
```

```
__attribute__((always_inline)) __STATIC_INLINE void __ISB(void)  
{  
    __ASM volatile ("isb 0xF":::"memory");  
}
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

```
*/
```

```
__attribute__((always_inline)) __STATIC_INLINE void __DSB(void)  
{  
    __ASM volatile ("dsb 0xF":::"memory");  
}
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before

and after the instruction, without ensuring their completion.

```
*/  
  
__attribute__((always_inline)) __STATIC_INLINE void __DMB(void)  
{  
    __ASM volatile ("dmb 0xF:::memory");  
}
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

```
\param [in]  value Value to reverse  
\return      Reversed value  
*/  
  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV(uint32_t value)  
{  
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 5)  
    return __builtin_bswap32(value);  
#else  
    uint32_t result;  
  
    __ASM volatile ("rev %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));  
    return(result);  
#endif  
}  
  
/** \brief Reverse byte order (16 bit)
```

This function reverses the byte order in two unsigned short values.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV16(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("rev16 %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value) );
    return(result);
}

/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE int32_t __REVSH(int32_t value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    return (short)__builtin_bswap16(value);
#else
    uint32_t result;
```

```
__ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value) );
return(result);
#endif
}
```

/** \brief Rotate Right in unsigned value (32 bit)

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```
\param [in]  value Value to rotate
\param [in]  value Number of Bits to rotate
\return      Rotated value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __ROR(uint32_t op1, uint32_t op2)
{
    return (op1 >> op2) | (op1 << (32 - op2));
}
```

/** \brief Breakpoint

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

\param [in] value is ignored by the processor.

If required, a debugger can use it to store additional information about the breakpoint.

```
*/  
#define __BKPT(value)           __ASM volatile ("bkpt "#value)
```

```
/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

```
\param [in]  value Value to reverse  
\return      Reversed value  
*/  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t value)  
{  
    uint32_t result;  
  
    #if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)  
        __ASM volatile ("rbit %0, %1" : "=r" (result) : "r" (value));  
    #else  
        int32_t s = 4 /*sizeof(v)*/ * 8 - 1; // extra shift needed at end  
  
        result = value;          // r will be reversed bits of v; first get LSB of v  
        for (value >>= 1; value; value >>= 1)  
        {  
            result <<= 1;  
            result |= value & 1;  
            s--;  
        }  
        result <<= s;          // shift when v's highest bits are zero  
    #endif
```

```
    return(result);  
}  
  
/** \brief Count leading zeros
```

This function counts the number of leading zeros of a data value.

```
\param [in] value Value to count the leading zeros  
\return number of leading zeros in value  
*/  
  
#define __CLZ      __builtin_clz
```

```
#if  (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
```

```
/** \brief LDR Exclusive (8 bit)
```

This function executes a exclusive LDR instruction for 8 bit value.

```
\param [in]  ptr Pointer to data  
\return     value of type uint8_t at (*ptr)  
*/  
  
__attribute__((always_inline)) __STATIC_INLINE uint8_t __LDREXB(volatile uint8_t *addr)  
{  
    uint32_t result;  
  
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)  
    __ASM volatile ("ldrexb %0, %1" : "=r" (result) : "Q" (*addr));
```

```

#else

/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
accepted by assembler. So has to use following less efficient pattern.

*/
__ASM volatile ("ldrexb %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );
#endif

return ((uint8_t) result); /* Add explicit type cast here */

}

```

/** \brief LDR Exclusive (16 bit)

This function executes a exclusive LDR instruction for 16 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint16_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint16_t __LDREXH(volatile uint16_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldreh %0, %1" : "=r" (result) : "Q" (*addr) );
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
accepted by assembler. So has to use following less efficient pattern.

*/
__ASM volatile ("ldrexb %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );
#endif

```

```
    return ((uint16_t) result); /* Add explicit type cast here */  
}
```

```
/** \brief LDR Exclusive (32 bit)
```

This function executes a exclusive LDR instruction for 32 bit values.

```
\param [in]  ptr  Pointer to data  
\return     value of type uint32_t at (*ptr)  
*/  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __LDREXW(volatile uint32_t *addr)  
{  
    uint32_t result;  
  
    __ASM volatile ("ldrex %0, %1" : "=r" (result) : "Q" (*addr));  
    return(result);  
}
```

```
/** \brief STR Exclusive (8 bit)
```

This function executes a exclusive STR instruction for 8 bit values.

```
\param [in]  value  Value to store  
\param [in]  ptr  Pointer to location  
\return     0  Function succeeded  
\return     1  Function failed  
*/
```

```

__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXB(uint8_t value, volatile uint8_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value));
    return(result);
}

```

/** \brief STR Exclusive (16 bit)

This function executes a exclusive STR instruction for 16 bit values.

```

\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXH(uint16_t value, volatile uint16_t
*addr)
{
    uint32_t result;

    __ASM volatile ("stre %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value));
    return(result);
}

```

```
/** \brief STR Exclusive (32 bit)
```

This function executes a exclusive STR instruction for 32 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
\return 0 Function succeeded  
\return 1 Function failed  
*/  
attribute((always_inline)) STATIC_INLINE uint32_t __STREXW(uint32_t value, volatile uint32_t  
*addr)  
{  
    uint32_t result;  
  
    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" (value));  
    return(result);  
}
```

```
/** \brief Remove the exclusive lock
```

This function removes the exclusive lock which is created by LDREX.

```
*/  
attribute((always_inline)) STATIC_INLINE void __CLREX(void)  
{  
    __ASM volatile ("clrex" :::"memory");  
}
```

```
/** \brief Signed Saturate
```

This function saturates a signed value.

```
\param [in] value Value to be saturated  
\param [in] sat Bit position to saturate to (1..32)  
\return Saturated value  
*/  
  
#define __SSAT(ARG1,ARG2) \  
{           \  
    uint32_t __RES, __ARG1 = (ARG1); \  
    __ASM ("ssat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \  
    __RES; \  
}
```

```
/** \brief Unsigned Saturate
```

This function saturates an unsigned value.

```
\param [in] value Value to be saturated  
\param [in] sat Bit position to saturate to (0..31)  
\return Saturated value  
*/  
  
#define __USAT(ARG1,ARG2) \  
{           \  
    uint32_t __RES, __ARG1 = (ARG1); \  
    __ASM ("usat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \  
}
```

```
__RES; \
})
```

```
/** \brief Rotate Right with Extend (32 bit)
```

This function moves each bit of a bitstring right by one bit.

The carry input is shifted in at the left end of the bitstring.

```
\param [in]  value  Value to rotate
\return      Rotated value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RRX(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("rrx %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
}
```

```
/** \brief LDRT Unprivileged (8 bit)
```

This function executes a Unprivileged LDRT instruction for 8 bit value.

```
\param [in]  ptr  Pointer to data
\return      value of type uint8_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint8_t __LDRBT(volatile uint8_t *addr)
```

```

{

    uint32_t result;

#ifndef __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrbt %0, %1" : "=r" (result) : "Q" (*addr));
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.

*/
    __ASM volatile ("ldrbt %0, [%1]" : "=r" (result) : "r" (addr) : "memory");
#endif

    return ((uint8_t) result); /* Add explicit type cast here */
}

/** \brief LDRT Unprivileged (16 bit)

```

This function executes a Unprivileged LDRT instruction for 16 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint16_t at (*ptr)

*/
__attribute__((always_inline)) __STATIC_INLINE uint16_t __LDRHT(volatile uint16_t *addr)
{
    uint32_t result;

#ifndef __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrht %0, %1" : "=r" (result) : "Q" (*addr));
#else

```

```

/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.

*/
__ASM volatile ("ldrht %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );
#endif

return ((uint16_t) result); /* Add explicit type cast here */
}

```

/** \brief LDRT Unprivileged (32 bit)

This function executes a Unprivileged LDRT instruction for 32 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint32_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __LDRT(volatile uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrt %0, %1" : "=r" (result) : "Q" (*addr) );
    return(result);
}

```

/** \brief STRT Unprivileged (8 bit)

This function executes a Unprivileged STRT instruction for 8 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
*/
__attribute__((always_inline)) __STATIC_INLINE void __STRBT(uint8_t value, volatile uint8_t *addr)
{
    __ASM volatile ("strbt %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value));
}
```

```
/** \brief STRT Unprivileged (16 bit)
```

This function executes a Unprivileged STRT instruction for 16 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
*/
__attribute__((always_inline)) __STATIC_INLINE void __STRHT(uint16_t value, volatile uint16_t *addr)
{
    __ASM volatile ("strht %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value));
}
```

```
/** \brief STRT Unprivileged (32 bit)
```

This function executes a Unprivileged STRT instruction for 32 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
*/
```

```
__attribute__((always_inline)) __STATIC_INLINE void __STRT(uint32_t value, volatile uint32_t *addr)
{
    __ASM volatile ("strt %1, %0" :"=Q" (*addr) : "r" (value));
}
```

```
#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */
```

```
#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>
```

```
#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>
```

```
#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING carm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
*/
```

```
#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
```

```
#include <cmsis_csm.h>

#endif

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */

#endif /* __CORE_CMINSTR_H */
*****/*
* @file core_cm0plus.h
* @brief CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*
*****
/* Copyright (c) 2009 - 2015 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif

#ifndef __CORE_CM0PLUS_H_GENERIC
#define __CORE_CM0PLUS_H_GENERIC

#ifdef __cplusplus
extern "C" {

#endif

/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions
CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.

*/

```
*****  
*      CMSIS definitions  
*****/  
/** \ingroup Cortex-M0+  
@{  
*/  
  
/* CMSIS CM0P definitions */  
  
#define __CMOPLUS_CMSIS_VERSION_MAIN (0x04)          /*!< [31:16] CMSIS HAL main  
version */  
  
#define __CMOPLUS_CMSIS_VERSION_SUB (0x00)           /*!< [15:0] CMSIS HAL sub version  
*/  
  
#define __CMOPLUS_CMSIS_VERSION  ((__CMOPLUS_CMSIS_VERSION_MAIN << 16) | \  
                                __CMOPLUS_CMSIS_VERSION_SUB)    /*!< CMSIS HAL version number */  
  
#define __CORTEX_M      (0x00)                      /*!< Cortex-M Core */
```

```

#if defined (__CC_ARM )
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE    __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined ( __GNUC__ )
#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined ( __ICCARM__ )
#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

#elif defined ( __TMS470__ )
#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#elif defined ( __TASKING__ )
#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

#elif defined ( __CSMC__ )
#define __packed
#define __ASM      __asm           /*!< asm keyword for COSMIC Compiler */

```

```

#define __INLINE      inline           /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */

#define __STATIC_INLINE static inline

#endif

/** __FPU_USED indicates whether an FPU is used or not.
    This core does not support an FPU at all
*/
#define __FPU_USED    0

#if defined ( __CC_ARM )

#if defined __TARGET_FPU_VFP
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __GNUC__ )

#if defined (__VFP_FP__) && !defined(__SOFTFP__)
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __ICCARM__ )

#if defined __ARMVFP__
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __TMS470__ )

#if defined __TI__VFP_SUPPORT__
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

```

```
#endif

#if defined ( __TASKING__ )
#if defined __FPU_VFP__
#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#if defined ( __CSMC__ )           /* Cosmic */
#if ( __CSMC__ & 0x400)          // FPU present for parser
#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif
#endif

#endif

#include <stdint.h>           /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access        */
#include <core_cmFunc.h>         /* Core Function Access           */

#endif /* __cplusplus

#endif /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

#endif /* __cplusplus
```

```
extern "C" {

#endif

/* check device defines and use defaults */

#if defined __CHECK_DEVICE_DEFINES

#ifndef __CM0PLUS_REV
#define __CM0PLUS_REV      0x0000
#warning "__CM0PLUS_REV not defined in device header file; using default!"
#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT      0
#warning "__MPU_PRESENT not defined in device header file; using default!"
#endif

#ifndef __VTOR_PRESENT
#define __VTOR_PRESENT      0
#warning "__VTOR_PRESENT not defined in device header file; using default!"
#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      2
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig  0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif
```

```

#endif

/* IO definitions (access restrictions to peripheral registers) */


\defgroup CMSIS_glob_defs CMSIS Global Defines

<strong>IO Type Qualifiers</strong> are used

\li to specify the access to peripheral variables.

\li for automatic generation of peripheral register debug information.

*/


#ifndef __cplusplus

#define __I volatile /*!< Defines 'read only' permissions */
#define __I volatile const /*!< Defines 'read only' permissions */
#define __O volatile /*!< Defines 'write only' permissions */
#define __IO volatile /*!< Defines 'read / write' permissions */

/*}@} end of group Cortex-M0+ */


*****
*      Register Abstraction


Core Register contain:


- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register

```

- Core MPU Register

```
*****/
```

```
/** \defgroup CMSIS_core_register Defines and Type Definitions
 \brief Type definitions and defines for Cortex-M processor based devices.

 */
```

```
/*\ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.

 @{

 */
```

```
/** \brief Union type to access the Application Program Status Register (APSR).
```

```
*/
```

```
typedef union
{
    struct
    {
        uint32_t _reserved0:28;      /*!< bit: 0..27 Reserved          */
        uint32_t V:1;              /*!< bit: 28 Overflow condition code flag   */
        uint32_t C:1;              /*!< bit: 29 Carry condition code flag     */
        uint32_t Z:1;              /*!< bit: 30 Zero condition code flag      */
        uint32_t N:1;              /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access          */
    uint32_t w;                /*!< Type used for word access            */
} APSR_Type;
```

```
/* APSR Register Definitions */
```

```
#define APSR_N_Pos           31           /*!< APSR: N Position */
```

```
#define APSR_N_Msk          (1UL << APSR_N_Pos)           /*!< APSR: N Mask */
```

```
#define APSR_Z_Pos          30                           /*!< APSR: Z Position */
```

```
#define APSR_Z_Msk          (1UL << APSR_Z_Pos)           /*!< APSR: Z Mask */
```

```
#define APSR_C_Pos          29                           /*!< APSR: C Position */
```

```
#define APSR_C_Msk          (1UL << APSR_C_Pos)           /*!< APSR: C Mask */
```

```
#define APSR_V_Pos          28                           /*!< APSR: V Position */
```

```
#define APSR_V_Msk          (1UL << APSR_V_Pos)           /*!< APSR: V Mask */
```

```
/** \brief Union type to access the Interrupt Program Status Register (IPSR).
```

```
*/
```

```
typedef union
```

```
{
```

```
    struct
```

```
{
```

```
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
```

```
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
```

```
    } b;                   /*!< Structure used for bit access */
```

```
    uint32_t w;             /*!< Type used for word access */
```

```
} IPSR_Type;
```

```
/* IPSR Register Definitions */
```

```
#define IPSR_ISR_Pos        0                           /*!< IPSR: ISR Position */
```

```
#define IPSR_ISR_Msk        (0x1FFUL /*<< IPSR_ISR_Pos*/)      /*!< IPSR: ISR Mask */
```

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:15;   /*!< bit: 9..23 Reserved */
        uint32_t T:1;             /*!< bit: 24 Thumb bit (read 0) */
        uint32_t _reserved1:3;    /*!< bit: 25..27 Reserved */
        uint32_t V:1;             /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;             /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;             /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;             /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

```

/ xPSR Register Definitions */*

#define xPSR_N_Pos	31	/*!< xPSR: N Position */
#define xPSR_N_Msk	(1UL << xPSR_N_Pos)	/*!< xPSR: N Mask */
#define xPSR_Z_Pos	30	/*!< xPSR: Z Position */
#define xPSR_Z_Msk	(1UL << xPSR_Z_Pos)	/*!< xPSR: Z Mask */
#define xPSR_C_Pos	29	/*!< xPSR: C Position */
#define xPSR_C_Msk	(1UL << xPSR_C_Pos)	/*!< xPSR: C Mask */
#define xPSR_V_Pos	28	/*!< xPSR: V Position */


```
#define CONTROL_nPRIV_Msk          (1UL /*<< CONTROL_nPRIV_Pos*/)      /*!< CONTROL:  
nPRIV Mask */
```

```
/*@} end of group CMSIS_CORE */
```

```
/** \ingroup CMSIS_core_register  
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)  
\brief Type definitions for the NVIC Registers  
{  
*/
```

```
/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).  
*/  
typedef struct  
{  
    __IO uint32_t ISER[1];           /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */  
    uint32_t RESERVED0[31];  
    __IO uint32_t ICER[1];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */  
    uint32_t RSERVED1[31];  
    __IO uint32_t ISPR[1];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */  
    uint32_t RESERVED2[31];  
    __IO uint32_t ICPR[1];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */  
    uint32_t RESERVED3[31];  
    uint32_t RESERVED4[64];  
    __IO uint32_t IP[8];            /*!< Offset: 0x300 (R/W) Interrupt Priority Register */  
} NVIC_Type;
```

```
/*@} end of group CMSIS_NVIC */
```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_SCB System Control Block (SCB)
 \brief Type definitions for the System Control Block Registers

@{

*/
/* Structure type to access the System Control Block (SCB).
 */
typedef struct
{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;           /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
    __I uint32_t VTOR;            /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
    #if (__VTOR_PRESENT == 1)
    __IO uint32_t RESERVED0;
    #else
    uint32_t RESERVED0;
    #endif
    __IO uint32_t AIRCR;          /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control Register */
    __IO uint32_t SCR;             /*!< Offset: 0x010 (R/W) System Control Register          */
    __IO uint32_t CCR;             /*!< Offset: 0x014 (R/W) Configuration Control Register   */
    uint32_t RESERVED1;
    __IO uint32_t SHP[2];          /*!< Offset: 0x01C (R/W) System Handlers Priority Registers. [0] is RESERVED */
    __IO uint32_t SHCSR;           /*!< Offset: 0x024 (R/W) System Handler Control and State Register */
}

} SCB_Type;

```

```

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24          /*!< SCB CPUID:  
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk     (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)    /*!<  
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos        20          /*!< SCB CPUID: VARIANT Position  
*/

#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos)    /*!< SCB CPUID:  
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos   16          /*!< SCB CPUID:  
ARCHITECTURE Position */

#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)  /*!< SCB  
CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos         4           /*!< SCB CPUID: PARTNO Position  
*/

#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos)    /*!< SCB  
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos       0           /*!< SCB CPUID: REVISION Position  
*/

#define SCB_CPUID_REVISION_Msk      (0xFUL /*<< SCB_CPUID_REVISION_Pos*/)  /*!< SCB  
CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos     31          /*!< SCB ICSR: NMIPENDSET  
Position */

#define SCB_ICSR_NMIPENDSET_Msk     (1UL << SCB_ICSR_NMIPENDSET_Pos)    /*!< SCB ICSR:  
NMIPENDSET Mask */

```

#define SCB_ICSR_PENDSVSET_Pos	28	/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk	(1UL << SCB_ICSR_PENDSVSET_Pos)	/*!< SCB ICSR: PENDSVSET Mask */
#define SCB_ICSR_PENDSVCLR_Pos	27	/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */
#define SCB_ICSR_VECTPENDING_Pos	12	/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk	(0x1FFUL << SCB_ICSR_VECTPENDING_Pos)	/*!< SCB ICSR: VECTPENDING Mask */

```

#define SCB_ICSR_VECTACTIVE_Pos      0           /*!< SCB ICSR: VECTACTIVE Position */
#define SCB_ICSR_VECTACTIVE_Msk     (0x1FFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/) /*!< SCB ICSR: VECTACTIVE Mask */

#if (__VTOR_PRESENT == 1)
/* SCB Interrupt Control State Register Definitions */

#define SCB_VTOR_TBLOFF_Pos        8           /*!< SCB VTOR: TBLOFF Position */
#define SCB_VTOR_TBLOFF_Msk       (0xFFFFFUL << SCB_VTOR_TBLOFF_Pos) /*!< SCB VTOR: TBLOFF Mask */

#endif

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos      16          /*!< SCB AIRCR: VECTKEY Position */
#define SCB_AIRCR_VECTKEY_Msk     (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos) /*!< SCB AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos    16          /*!< SCB AIRCR: VECTKEYSTAT Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk   (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos) /*!< SCB AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos     15          /*!< SCB AIRCR: ENDIANESS Position */
#define SCB_AIRCR_ENDIANESS_Msk   (1UL << SCB_AIRCR_ENDIANESS_Pos) /*!< SCB AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos    2           /*!< SCB AIRCR: SYSRESETREQ Position */
#define SCB_AIRCR_SYSRESETREQ_Msk   (1UL << SCB_AIRCR_SYSRESETREQ_Pos) /*!< SCB AIRCR: SYSRESETREQ Mask */

```

```

#define SCB_AIRCR_VECTCLRACTIVE_Pos      1          /*!< SCB AIRCR: VECTCLRACTIVE Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk     (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)      /*!< SCB AIRCR: VECTCLRACTIVE Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos         4          /*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk        (1UL << SCB_SCR_SEVONPEND_Pos)      /*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos        2          /*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk       (1UL << SCB_SCR_SLEEPDEEP_Pos)      /*!< SCB SCR: SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos      1          /*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk    (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos        9          /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk       (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR: STKALIGN Mask */

#define SCB_CCR_UNALIGN_TRP_Pos     3          /*!< SCB CCR: UNALIGN_TRP Position */
#define SCB_CCR_UNALIGN_TRP_Msk   (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB CCR: UNALIGN_TRP Mask */

/* SCB System Handler Control and State Register Definitions */

```

```
#define SCB_SHCSR_SVCALLPENDED_Pos    15           /*!< SCB SHCSR:  
SVCALLPENDED Position */  
  
#define SCB_SHCSR_SVCALLPENDED_Msk     (1UL << SCB_SHCSR_SVCALLPENDED_Pos)      /*!< SCB  
SHCSR: SVCALLPENDED Mask */
```

```
/*@} end of group CMSIS_SCB */
```

```
/** \ingroup CMSIS_core_register  
\\defgroup CMSIS_SysTick System Tick Timer (SysTick)  
\\brief Type definitions for the System Timer Registers.  
@{  
*/  
  
/** \\brief Structure type to access the System Timer (SysTick).  
*/  
  
typedef struct  
{  
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */  
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */  
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */  
    __I uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */  
} SysTick_Type;  
  
/* SysTick Control / Status Register Definitions */  
  
#define SysTick_CTRL_COUNTFLAG_Pos    16           /*!< SysTick CTRL: COUNTFLAG  
Position */  
  
#define SysTick_CTRL_COUNTFLAG_Msk    (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick  
CTRL: COUNTFLAG Mask */
```

```

#define SysTick_CTRL_CLKSOURCE_Pos      2          /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk     (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos       1          /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk      (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos        0          /*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk       (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)      /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos       0          /*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk      (0xFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/)  /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos       0          /*!< SysTick VAL: CURRENT Position */
#define SysTick_VAL_CURRENT_Msk      (0xFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/)  /*!< SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos      31         /*!< SysTick CALIB: NOREF Position */
#define SysTick_CALIB_NOREF_Msk      (1UL << SysTick_CALIB_NOREF_Pos)      /*!< SysTick CALIB: NOREF Mask */

```

```

#define SysTick_CALIB_SKEW_Pos      30           /*!< SysTick CALIB: SKEW Position
*/
#define SysTick_CALIB_SKEW_Msk      (1UL << SysTick_CALIB_SKEW_Pos)    /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos     0            /*!< SysTick CALIB: TENMS Position
*/
#define SysTick_CALIB_TENMS_Msk     (0xFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

/*}@} end of group CMSIS_SysTick */

#if (__MPU_PRESENT == 1)
/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUs Memory Protection Unit (MPU)
\brief Type definitions for the Memory Protection Unit (MPU)

@{
*/

/** \brief Structure type to access the Memory Protection Unit (MPU).
*/
typedef struct
{
    __I uint32_t TYPE;          /*!< Offset: 0x000 (R/ ) MPU Type Register */
    __IO uint32_t CTRL;         /*!< Offset: 0x004 (R/W) MPU Control Register */
    __IO uint32_t RNR;          /*!< Offset: 0x008 (R/W) MPU Region RNRber Register */
    __IO uint32_t RBAR;         /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
    __IO uint32_t RASR;         /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register
*/
} MPU_Type;

```

```

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)    /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)    /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)  /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

#define MPU_CTRL_PRIVDEFENA_Pos   2           /*!< MPU CTRL: PRIVDEFENA
Position */
#define MPU_CTRL_PRIVDEFENA_Msk   (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos    1           /*!< MPU CTRL: HFNMIENA
Position */
#define MPU_CTRL_HFNMIENA_Msk    (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos       0           /*!< MPU CTRL: ENABLE Position */
#define MPU_CTRL_ENABLE_Msk       (1UL /*<< MPU_CTRL_ENABLE_Pos*/)  /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */

```

```

#define MPU_RNR_REGION_Pos      0           /*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk      (0xFFUL /*<< MPU_RNR_REGION_Pos*/)    /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos       8           /*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk       (0xFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos      4           /*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk      (1UL << MPU_RBAR_VALID_Pos)        /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos      0           /*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk      (0xFUL /*<< MPU_RBAR_REGION_Pos*/)  /*!< MPU
RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos       16          /*!< MPU RASR: MPU Region
Attribute field Position */
#define MPU_RASR_ATTRS_Msk       (0xFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos          28          /*!< MPU RASR: ATTRS.XN Position */
#define MPU_RASR_XN_Msk          (1UL << MPU_RASR_XN_Pos)        /*!< MPU RASR:
ATTRS.XN Mask */

#define MPU_RASR_AP_Pos          24          /*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk          (0x7UL << MPU_RASR_AP_Pos)      /*!< MPU RASR:
ATTRS.AP Mask */

```

```

#define MPU_RASR_TEX_Pos           19          /*!< MPU RASR: ATTRS.TEX Position */
*/
#define MPU_RASR_TEX_Msk          (0x7UL << MPU_RASR_TEX_Pos)      /*!< MPU RASR:
ATTRS.TEX Mask */

#define MPU_RASR_S_Pos             18          /*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk            (1UL << MPU_RASR_S_Pos)      /*!< MPU RASR: ATTRS.S
Mask */

#define MPU_RASR_C_Pos             17          /*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk            (1UL << MPU_RASR_C_Pos)      /*!< MPU RASR:
ATTRS.C Mask */

#define MPU_RASR_B_Pos             16          /*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk            (1UL << MPU_RASR_B_Pos)      /*!< MPU RASR:
ATTRS.B Mask */

#define MPU_RASR_SRD_Pos           8           /*!< MPU RASR: Sub-Region Disable
Position */
#define MPU_RASR_SRD_Msk          (0xFFUL << MPU_RASR_SRD_Pos)    /*!< MPU RASR:
Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos          1           /*!< MPU RASR: Region Size Field
Position */
#define MPU_RASR_SIZE_Msk         (0x1FUL << MPU_RASR_SIZE_Pos)    /*!< MPU RASR:
Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos         0          /*!< MPU RASR: Region enable bit
Position */
#define MPU_RASR_ENABLE_Msk        (1UL /*<< MPU_RASR_ENABLE_Pos*/)  /*!< MPU
RASR: Region enable bit Disable Mask */

```

```

/*@} end of group CMSIS_MPU */

#endif

/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug Core Debug Registers (CoreDebug)
\brief Cortex-M0+ Core Debug Registers (DCB registers, SHCSR, and DFSR)
are only accessible over DAP and not via processor. Therefore
they are not covered by the Cortex-M0 header file.

@{
*/
/*@} end of group CMSIS_CoreDebug */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_core_base Core Definitions
\brief Definitions for base addresses, unions, and structures.

@{
*/
/* Memory mapping of Cortex-M0+ Hardware */

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */
#define SysTick_BASE  (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */
#define NVIC_BASE    (SCS_BASE + 0x0100UL)       /*!< NVIC Base Address */
#define SCB_BASE     (SCS_BASE + 0x0D00UL)       /*!< System Control Block Base Address */

#define SCB          ((SCB_Type *) SCB_BASE)    /*!< SCB configuration struct */
#define SysTick      ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */
#define NVIC         ((NVIC_Type *) NVIC_BASE)   /*!< NVIC configuration struct */

```

```
#if (_MPU_PRESENT == 1)

#define MPU_BASE      (SCS_BASE + 0x0D90UL)           /*!< Memory Protection Unit */
#define MPU      ((MPU_Type     *) MPU_BASE) /*!< Memory Protection Unit */

#endif

/*@} */
```

```
*****
*      Hardware Abstraction Layer
Core Function Interface contains:
- Core NVIC Functions
- Core SysTick Functions
- Core Register Access Functions
*****/
```

/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/

```
/* ##### NVIC functions ##### */
```

/** \ingroup CMSIS_Core_FunctionInterface

```
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
```

\brief Functions that manage interrupts and exceptions via the NVIC.

```
@{
```

```
*/
```

```
/* Interrupt Priorities are WORD accessible only under ARMv6M */  
/* The following MACROS handle generation of the register offset and byte masks */  
  
#define _BIT_SHIFT(IRQn)      ( (((uint32_t)(int32_t)(IRQn))      ) & 0x03UL) * 8UL  
#define _SHP_IDX(IRQn)        ( (((((uint32_t)(int32_t)(IRQn)) & 0x0FUL)-8UL) >> 2UL) )  
#define _IP_IDX(IRQn)         ( (((uint32_t)(int32_t)(IRQn))           >> 2UL) )
```

```
/** \brief Enable External Interrupt
```

The function enables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]    IRQn  External interrupt number. Value cannot be negative.  
*/  
  
__STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]    IRQn  External interrupt number. Value cannot be negative.  
*/  
  
__STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

*/

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
{
    return((uint32_t)((NVIC->ISPR[0] & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL :
    0UL);
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

\param [in] IRQn Interrupt number. Value cannot be negative.

*/

```
__STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ISPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ICPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
    if((int32_t)(IRQn) < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = ((uint32_t)(SCB->SHP[_SHP_IDX(IRQn)] & ~ (0xFFUL << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) << _BIT_SHIFT(IRQn)));
    }
}
```

```

    }

else {
    NVIC->IP[_IP_IDX(IRQn)] = ((uint32_t)(NVIC->IP[_IP_IDX(IRQn)] & ~(0xFFUL << _BIT_SHIFT(IRQn))) |
        (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) << _BIT_SHIFT(IRQn)));
}

}

```

*/** \brief Get Interrupt Priority*

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

\param [in] IRQn Interrupt number.

\return Interrupt Priority. Value is aligned automatically to the implemented priority bits of the microcontroller.

*/

```

__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{
    if((int32_t)(IRQn) < 0) {
        return((uint32_t)((SCB->SHP[_SHP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & (uint32_t)0xFFUL) >> (8 -
            __NVIC_PRIO_BITS));
    }
    else {
        return((uint32_t)((NVIC->IP[_IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & (uint32_t)0xFFUL) >> (8 -
            __NVIC_PRIO_BITS));
    }
}
```

```
}
```

```
/** \brief System Reset
```

The function initiates a system reset request to reset the MCU.

```
*/
```

```
_STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                /* Ensure all outstanding memory accesses included
                                              buffered write are completed before reset */

    SCB->AIRCR = ((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                    SCB_AIRCR_SYSRESETREQ_Msk);

    __DSB();                                /* Ensure completion of memory access */

    while(1) { __NOP(); }                   /* wait until reset */
}
```

```
/*@} end of CMSIS_Core_NVICFunctions */
```

```
/* ##### SysTick function
##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.

 @{
 */

```

```
#if (__Vendor_SysTickConfig == 0)
```

```
/** \brief System Tick Configuration
```

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.

Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.

\return 1 Function failed.

\note When the variable **__Vendor_SysTickConfig** is set to 1, then the function **SysTick_Config** is not included. In this case, the file **<i>device</i>.h** must contain a vendor-specific implementation of this function.

```
*/
```

```
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
```

```
{
```

```
if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) {return (1UL); /* Reload value impossible */}
```

```
SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */
```

```
NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
```

```
SysTick->VAL = 0UL; /* Load the SysTick Counter Value */
```

```
SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
```

```
    SysTick_CTRL_TICKINT_Msk |
```

```
    SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */
```

```
    return (0UL);                                /* Function successful */  
}  
  
#endif
```

```
/*}@} end of CMSIS_Core_SysTickFunctions */
```

```
#ifdef __cplusplus  
}  
#endif
```

```
#endif /* __CORE_CM0PLUS_H_DEPENDANT */
```

```
#endif /* __CMSIS_GENERIC */  
/*  
** #####  
** Processors:      MKL25Z128FM4  
**                  MKL25Z128FT4  
**                  MKL25Z128LH4  
**                  MKL25Z128VLK4  
**  
** Compilers:       Keil ARM C/C++ Compiler  
**                  Freescale C/C++ for Embedded ARM  
**                  GNU C Compiler  
**                  GNU C Compiler - CodeSourcery Sourcery G++  
**                  IAR ANSI C/C++ Compiler for ARM
```

**

** Reference manual: KL25P80M48SFORM, Rev.3, Sep 2012

** Version: rev. 2.5, 2015-02-19

** Build: b150220

**

** Abstract:

** Provides a system configuration function and a global variable that

** contains the system frequency. It configures the device and initializes

** the oscillator (PLL) that is part of the microcontroller device.

**

** Copyright (c) 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com
** mail: support@freescale.com

**

** Revisions:

** - rev. 1.0 (2012-06-13)
** Initial version.
** - rev. 1.1 (2012-06-21)
** Update according to reference manual rev. 1.
** - rev. 1.2 (2012-08-01)
** Device type UARTLP changed to UART0.
** - rev. 1.3 (2012-10-04)
** Update according to reference manual rev. 3.
** - rev. 1.4 (2012-11-22)
** MCG module - bit LOLS in MCG_S register renamed to LOLS0.
** NV registers - bit EZPORT_DIS in NV_FOPT register removed.
** - rev. 1.5 (2013-04-05)
** Changed start of doxygen comment.
** - rev. 2.0 (2013-10-29)
** Register accessor macros added to the memory map.
** Symbols for Processor Expert memory map compatibility added to the memory map.

```
** Startup file for gcc has been updated according to CMSIS 3.2.  
** System initialization updated.  
** - rev. 2.1 (2014-07-16)  
** Module access macro module_BASES replaced by module_BASE_PTRS.  
** System initialization and startup updated.  
** - rev. 2.2 (2014-08-22)  
** System initialization updated - default clock config changed.  
** - rev. 2.3 (2014-08-28)  
** Update of startup files - possibility to override DefaultISR added.  
** - rev. 2.4 (2014-10-14)  
** Interrupt INT_LPTimer renamed to INT_LPTMR0.  
** - rev. 2.5 (2015-02-19)  
** Renamed interrupt vector LLW to LLWU.  
**  
** #####  
*/
```

```
/*!  
* @file MKL25Z4  
* @version 2.5  
* @date 2015-02-19  
* @brief Device specific configuration file for MKL25Z4 (header file)  
*  
* Provides a system configuration function and a global variable that contains  
* the system frequency. It configures the device and initializes the oscillator  
* (PLL) that is part of the microcontroller device.  
*/
```

```
#ifndef SYSTEM_MKL25Z4_H_
```

```
#define SYSTEM_MKL25Z4_H_          /**< Symbol preventing repeated inclusion */
```

```
#ifdef __cplusplus
```

```
extern "C" {
```

```
#endif
```

```
#include <stdint.h>
```

```
#ifndef DISABLE_WDOG
```

```
#define DISABLE_WDOG      1
```

```
#endif
```

```
/* MCG mode constants */
```

```
#define MCG_MODE_FEI      0U
```

```
#define MCG_MODE_FBI      1U
```

```
#define MCG_MODE_BLPI     2U
```

```
#define MCG_MODE_FEE      3U
```

```
#define MCG_MODE_FBE      4U
```

```
#define MCG_MODE_BLPE     5U
```

```
#define MCG_MODE_PBE      6U
```

```
#define MCG_MODE_PEE      7U
```

```
/* Predefined clock setups
```

```
0 ... Default part configuration
```

Multipurpose Clock Generator (MCG) in FEI mode.

Reference clock source for MCG module: Slow internal reference clock

Core clock = 20.97152MHz

Bus clock = 20.97152MHz

1 ... Maximum achievable clock frequency configuration

Multipurpose Clock Generator (MCG) in PEE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 48MHz

Bus clock = 24MHz

2 ... Chip internally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPI mode.

Reference clock source for MCG module: Fast internal reference clock

Core clock = 4MHz

Bus clock = 0.8MHz

3 ... Chip externally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 4MHz

Bus clock = 1MHz

4 ... USB clock setup

Multipurpose Clock Generator (MCG) in PEE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 48MHz

Bus clock = 24MHz

*/

/* Define clock source values */

```

#define CPU_XTAL_CLK_HZ      8000000u      /* Value of the external crystal or oscillator clock
frequency in Hz */

#define CPU_INT_SLOW_CLK_HZ   32768u       /* Value of the slow internal oscillator clock
frequency in Hz */

#define CPU_INT_FAST_CLK_HZ   4000000u     /* Value of the fast internal oscillator clock
frequency in Hz */

/* RTC oscillator setting */

/* Low power mode enable */

/* SMC_PMPROT: AVLP=1,ALLS=1,AVLLS=1 */

#define SYSTEM_SMC_PMPROT_VALUE 0x2AU      /* SMC_PMPROT */

/* Internal reference clock trim */

/* #undef SLOW_TRIM_ADDRESS */           /* Slow oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef SLOW_FINE_TRIM_ADDRESS */      /* Slow oscillator not trimmed. Commented out
for MISRA compliance. */

/* #undef FAST_TRIM_ADDRESS */          /* Fast oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef FAST_FINE_TRIM_ADDRESS */      /* Fast oscillator not trimmed. Commented out
for MISRA compliance. */

#ifndef CLOCK_SETUP

#if (CLOCK_SETUP == 0)

#define DEFAULT_SYSTEM_CLOCK    20971520u    /* Default System clock value */

#define MCG_MODE                MCG_MODE_FEI /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE    0x06U        /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFS0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE    0x24U        /* MCG_C2 */

```

```

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOC50=0 */

#define SYSTEM_MCG_SC_VALUE     0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE     0x00U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE     0x00U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE    0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE  0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=0 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x00U      /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE   0x000C0000U /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE   0x01000000U /* SIM_SOPT2 */

#if (CLOCK_SETUP == 1)

#define DEFAULT_SYSTEM_CLOCK     48000000u /* Default System clock value */

#define MCG_MODE                 MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE     0x1AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSO=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE     0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE     0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOC50=0 */

#define SYSTEM_MCG_SC_VALUE     0x00U      /* MCG_SC */

```

```

/* MCG_C5: PLLCLKEN0=0,PLLSTENO=0,PRDIV0=1 */
#define SYSTEM_MCG_C5_VALUE      0x01U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE      0x40U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */
#define SYSTEM_SIM_CLKDIV1_VALUE  0x10010000U    /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U    /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 2)

#define DEFAULT_SYSTEM_CLOCK      4000000u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_BLPI /* Clock generator mode */

/* MCG_C1: CLKS=1,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE       0x46U      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFS0=1,LP=1,IRCS=1 */
#define SYSTEM_MCG_C2_VALUE       0x27U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE       0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCSS0=0 */
#define SYSTEM_MCG_SC_VALUE       0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTENO=0,PRDIV0=0 */
#define SYSTEM_MCG_C5_VALUE       0x00U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE       0x00U      /* MCG_C6 */

```

```

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE      0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=4 */

#define SYSTEM_SIM_CLKDIV1_VALUE  0x00040000U    /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE   0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE   0x02000000U    /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 3)

#define DEFAULT_SYSTEM_CLOCK     4000000u      /* Default System clock value */

#define MCG_MODE                 MCG_MODE_BLPE /* Clock generator mode */

/* MCG_C1: CLKS=2,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE     0x9AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSS0=1,LP=1,IRCS=1 */

#define SYSTEM_MCG_C2_VALUE     0x27U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE     0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=1,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE     0x02U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE     0x00U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE     0x00U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

```

```

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=3 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x10030000U /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE 0x02000000U /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 4)

#define DEFAULT_SYSTEM_CLOCK 48000000U /* Default System clock value */

#define MCG_MODE MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE 0x1AU /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSTEN=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE 0x24U /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE 0x00U /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCSD0=0 */

#define SYSTEM_MCG_SC_VALUE 0x00U /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=1 */

#define SYSTEM_MCG_C5_VALUE 0x01U /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE 0x40U /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE 0x80U /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE 0x00U /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x10010000U /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */

```

```

/* SIM_SOPT2: UARTOSRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U    /* SIM_SOPT2 */

#endif
#else
#define DEFAULT_SYSTEM_CLOCK      20971520u    /* Default System clock value */
#endif

/***
 * @brief System clock frequency (core clock)
 *
 * The system clock frequency supplied to the SysTick timer and the processor
 * core clock. This variable can be used by the user application to setup the
 * SysTick timer or configure other parameters. It may also be used by debugger to
 * query the frequency of the debug timer or configure the trace clock speed
 * SystemCoreClock is initialized with a correct predefined value.
 */
extern uint32_t SystemCoreClock;

/***
 * @brief Setup the microcontroller system.
 *
 * Typically this function configures the oscillator (PLL) that is part of the
 * microcontroller device. For systems with variable clock speed it also updates
 * the variable SystemCoreClock. SystemInit is called from startup_device file.
 */
void SystemInit (void);

/**

```

```

* @brief Updates the SystemCoreClock variable.

*
* It must be called whenever the core clock is changed during program
* execution. SystemCoreClockUpdate() evaluates the clock register settings and calculates
* the current core clock.

*/
void SystemCoreClockUpdate (void);

#ifndef __cplusplus
}

#endif /* #if !defined(SYSTEM_MKL25Z4_H_) */

//*******************************************************************************/

* @file core_cm4.h
* @brief CMSIS Cortex-M4 Core Peripheral Access Layer Header File
* @version V3.30
* @date 24. February 2014
*
* @note
*
//*******************************************************************************/

/* Copyright (c) 2009 - 2014 ARM LIMITED

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )  
#pragma system_include /* treat file as system include file for MISRA check */  
#endif
```

```
#ifndef __CORE_CM4_H_GENERIC  
#define __CORE_CM4_H_GENERIC
```

```
#ifdef __cplusplus
```

```
extern "C" {

#endif

/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions

CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.

*/

```
*****
```

* CMSIS definitions

```
*****
```

```
/** \ingroup Cortex_M4
```

```
@{
```

```
*/
```

```
/* CMSIS CM4 definitions */
```

```
#define __CM4_CMSIS_VERSION_MAIN (0x03)          /*!< [31:16] CMSIS HAL main version
*/
```

```
#define __CM4_CMSIS_VERSION_SUB (0x20)           /*!< [15:0] CMSIS HAL sub version */

#define __CM4_CMSIS_VERSION   ((__CM4_CMSIS_VERSION_MAIN << 16) | \
```

```

    __CM4_CMSIS_VERSION_SUB      ) /*!< CMSIS HAL version number      */

#define __CORTEX_M          (0x04)           /*!< Cortex-M Core          */

#if defined (__CC_ARM)
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)
#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)
#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

#elif defined (__TMS470__)
#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#elif defined (__TASKING__)
#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

```

```

#define __packed
#define __ASM      _asm          /*!< asm keyword for COSMIC Compiler */
#define __INLINE    inline        /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */
#define __STATIC_INLINE static inline

#endif

/** __FPU_USED indicates whether an FPU is used or not. For this, __FPU_PRESENT has to be checked
prior to making use of FPU specific registers and functions.

*/
#if defined ( __CC_ARM )
#if defined __TARGET_FPU_VFP
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED    0
#endif
#else
#define __FPU_USED    0
#endif

#endif

#if defined ( __GNUC__ )
#if defined (__VFP_FP__) && !defined(__SOFTFP__)
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1

```

```
#else

    #warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"

#define __FPU_USED    0

#endif

#ifndef __FPU_USED
#define __FPU_USED    0
#endif

#elif defined ( __ICCARM__ )

#if defined __ARMVFP__

#if (__FPU_PRESENT == 1)

#define __FPU_USED    1

#else

    #warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"

#define __FPU_USED    0

#endif

#ifndef __FPU_USED
#define __FPU_USED    0
#endif

#endif

#elif defined ( __TMS470__ )

#if defined __TI_VFP_SUPPORT__

#if (__FPU_PRESENT == 1)

#define __FPU_USED    1

#else

    #warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"

#define __FPU_USED    0
```

```
#endif

#else

#define __FPU_USED    0

#endif

#elif defined ( __TASKING__ )

#if defined __FPU_VFP__

#if (__FPU_PRESENT == 1)

#define __FPU_USED    1

#else

#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

#define __FPU_USED    0

#endif

#else

#define __FPU_USED    0

#endif

#endif

#elif defined ( __CSMC__ )          /* Cosmic */

#if ( __CSMC__ & 0x400)           // FPU present for parser

#if (__FPU_PRESENT == 1)

#define __FPU_USED    1

#else

#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

#define __FPU_USED    0

#endif

#else

#define __FPU_USED    0

#endif

#endif
```

```
#include <stdint.h>          /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access        */
#include <core_cmFunc.h>         /* Core Function Access          */
#include <core_cm4_simd.h>        /* Compiler specific SIMD Intrinsics */

#endif /* __CORE_CM4_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM4_H_DEPENDANT
#define __CORE_CM4_H_DEPENDANT

/* check device defines and use defaults */

#if defined __CHECK_DEVICE_DEFINES

#ifndef __CM4_REV
#define __CM4_REV      0x0000
#warning "__CM4_REV not defined in device header file; using default!"
#endif

#ifndef __FPU_PRESENT
#define __FPU_PRESENT   0
#warning "__FPU_PRESENT not defined in device header file; using default!"
#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT    0
#warning "__MPU_PRESENT not defined in device header file; using default!"
#endif


```

```

#ifndef __NVIC_PRIO_BITS

#define __NVIC_PRIO_BITS      4
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"

#endif


#ifndef __Vendor_SysTickConfig

#define __Vendor_SysTickConfig  0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"

#endif

#endif


/* IO definitions (access restrictions to peripheral registers) */

/**


\defgroup CMSIS_glob_defs CMSIS Global Defines

<strong>IO Type Qualifiers</strong> are used

\li to specify the access to peripheral variables.

\li for automatic generation of peripheral register debug information.

*/
#endif

#ifndef __cplusplus

#define __I volatile      /*!< Defines 'read only' permissions */
#ifndef __I
#define __I volatile const /*!< Defines 'read only' permissions */
#endif

#define __O volatile      /*!< Defines 'write only' permissions */
#define __IO volatile     /*!< Defines 'read / write' permissions */

/*@} end of group Cortex_M4 */

```

```

/***********************/

*      Register Abstraction

Core Register contain:

- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core Debug Register
- Core MPU Register
- Core FPU Register

***** */

/** \defgroup CMSIS_core_register Defines and Type Definitions
 \brief Type definitions and defines for Cortex-M processor based devices.
 */

/* \ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.

 @{
 */

/** \brief Union type to access the Application Program Status Register (APSR).
 */

typedef union
{
    struct

```

```

{

#if (__CORTEX_M != 0x04)

    uint32_t _reserved0:27;      /*!< bit: 0..26 Reserved          */

#else

    uint32_t _reserved0:16;      /*!< bit: 0..15 Reserved          */
    uint32_t GE:4;              /*!< bit: 16..19 Greater than or Equal flags */
    uint32_t _reserved1:7;      /*!< bit: 20..26 Reserved          */

#endif

    uint32_t Q:1;                /*!< bit: 27 Saturation condition flag */
    uint32_t V:1;                /*!< bit: 28 Overflow condition code flag */
    uint32_t C:1;                /*!< bit: 29 Carry condition code flag */
    uint32_t Z:1;                /*!< bit: 30 Zero condition code flag */
    uint32_t N:1;                /*!< bit: 31 Negative condition code flag */

} b;                      /*!< Structure used for bit access */

uint32_t w;                  /*!< Type used for word access */

} APSR_Type;

```

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0.. 8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved          */

    } b;                      /*!< Structure used for bit access */

    uint32_t w;                  /*!< Type used for word access */

} IPSR_Type;

```

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
#ifndef __CORTEX_M != 0x04
        uint32_t _reserved0:15;   /*!< bit: 9..23 Reserved */
#else
        uint32_t _reserved0:7;    /*!< bit: 9..15 Reserved */
        uint32_t GE:4;           /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:4;    /*!< bit: 20..23 Reserved */
#endif
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

```

/** \brief Union type to access the Control Registers (CONTROL).

```

*/
typedef union
{
    struct
    {
        uint32_t nPRIV:1;           /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} CONTROL_Type;

```

```
/*@} end of group CMSIS_CORE */
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
@{
*/
```

```
/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[8];      /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */
    uint32_t RESERVED0[24];
    __IO uint32_t ICER[8];      /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
}
```

```

    uint32_t RSERVED1[24];

__IO uint32_t ISPR[8];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register      */

    uint32_t RESERVED2[24];

__IO uint32_t ICPR[8];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register     */

    uint32_t RESERVED3[24];

__IO uint32_t IABR[8];           /*!< Offset: 0x200 (R/W) Interrupt Active bit Register      */

    uint32_t RESERVED4[56];

__IO uint8_t IP[240];           /*!< Offset: 0x300 (R/W) Interrupt Priority Register (8Bit wide) */

    uint32_t RESERVED5[644];

__O uint32_t STIR;              /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register */

} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */

#define NVIC_STIR_INTID_Pos          0                  /*!< STIR: INTLINESNUM Position */

#define NVIC_STIR_INTID_Msk          (0x1FUL << NVIC_STIR_INTID_Pos)      /*!< STIR:
INTLINESNUM Mask */

/*@} end of group CMSIS_NVIC */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers

@{
*/

```

```

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct

```

```

{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
*/                          

    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
    __IO uint32_t AIRCR;         /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register   */
    __IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register          */
    __IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register    */
    __IO uint8_t SHP[12];        /*!< Offset: 0x018 (R/W) System Handlers Priority Registers (4-7, 8-11,
12-15) */
    __IO uint32_t SHCSR;         /*!< Offset: 0x024 (R/W) System Handler Control and State Register */
*/                          

    __IO uint32_t CFSR;          /*!< Offset: 0x028 (R/W) Configurable Fault Status Register */
*/                          

    __IO uint32_t HFSR;          /*!< Offset: 0x02C (R/W) HardFault Status Register          */
    __IO uint32_t DFSR;          /*!< Offset: 0x030 (R/W) Debug Fault Status Register        */
    __IO uint32_t MMFAR;         /*!< Offset: 0x034 (R/W) MemManage Fault Address Register    */
*/                          

    __IO uint32_t BFAR;          /*!< Offset: 0x038 (R/W) BusFault Address Register          */
    __IO uint32_t AFSR;          /*!< Offset: 0x03C (R/W) Auxiliary Fault Status Register    */
    __I uint32_t PFR[2];         /*!< Offset: 0x040 (R/ ) Processor Feature Register        */
    __I uint32_t DFR;           /*!< Offset: 0x048 (R/ ) Debug Feature Register          */
    __I uint32_t ADR;            /*!< Offset: 0x04C (R/ ) Auxiliary Feature Register        */
    __I uint32_t MMFR[4];        /*!< Offset: 0x050 (R/ ) Memory Model Feature Register    */
*/                          

    __I uint32_t ISAR[5];        /*!< Offset: 0x060 (R/ ) Instruction Set Attributes Register */
    uint32_t RESERVED0[5];       /* */

    __IO uint32_t CPACR;         /*!< Offset: 0x088 (R/W) Coprocessor Access Control Register */
*/
}

} SCB_Type;

```

```

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24          /*!< SCB CPUID:  
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk     (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos) /*!<  
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos        20          /*!< SCB CPUID: VARIANT Position  
*/

#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos) /*!< SCB CPUID:  
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos   16          /*!< SCB CPUID:  
ARCHITECTURE Position */

#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos) /*!<  
SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos         4           /*!< SCB CPUID: PARTNO Position  
*/

#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos) /*!< SCB  
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos       0           /*!< SCB CPUID: REVISION Position  
*/

#define SCB_CPUID_REVISION_Msk     (0xFUL << SCB_CPUID_REVISION_Pos) /*!< SCB CPUID:  
REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos    31          /*!< SCB ICSR: NMIPENDSET  
Position */

#define SCB_ICSR_NMIPENDSET_Msk    (1UL << SCB_ICSR_NMIPENDSET_Pos) /*!< SCB ICSR:  
NMIPENDSET Mask */

```

#define SCB_ICSR_PENDSVSET_Pos	28	/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk	(1UL << SCB_ICSR_PENDSVSET_Pos)	/*!< SCB ICSR: PENDSVSET Mask */
#define SCB_ICSR_PENDSVCLR_Pos	27	/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */
#define SCB_ICSR_VECTPENDING_Pos	12	/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk	(0x1FFUL << SCB_ICSR_VECTPENDING_Pos)	/*!< SCB ICSR: VECTPENDING Mask */

```

#define SCB_ICSR_RETTOBASE_Pos      11           /*!< SCB ICSR: RETTOBASE
Position */

#define SCB_ICSR_RETTOBASE_Msk     (1UL << SCB_ICSR_RETTOBASE_Pos)    /*!< SCB ICSR:
RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos    0            /*!< SCB ICSR: VECTACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk   (0x1FFUL << SCB_ICSR_VECTACTIVE_Pos)  /*!< SCB ICSR:
VECTACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */

#define SCB_VTOR_TBLOFF_Pos       7            /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk      (0xFFFFFFFFUL << SCB_VTOR_TBLOFF_Pos)  /*!< SCB
VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos     16           /*!< SCB AIRCR: VECTKEY Position
*/

#define SCB_AIRCR_VECTKEY_Msk    (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)  /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos 16           /*!< SCB AIRCR: VECTKEYSTAT
Position */

#define SCB_AIRCR_VECTKEYSTAT_Msk (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos) /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos   15           /*!< SCB AIRCR: ENDIANESS
Position */

#define SCB_AIRCR_ENDIANESS_Msk  (1UL << SCB_AIRCR_ENDIANESS_Pos)    /*!< SCB
AIRCR: ENDIANESS Mask */

```

```

#define SCB_AIRCR_PRIGROUP_Pos      8          /*!< SCB AIRCR: PRIGROUP
Position */

#define SCB_AIRCR_PRIGROUP_Msk     (7UL << SCB_AIRCR_PRIGROUP_Pos)    /*!< SCB
AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos   2          /*!< SCB AIRCR: SYSRESETREQ
Position */

#define SCB_AIRCR_SYSRESETREQ_Msk   (1UL << SCB_AIRCR_SYSRESETREQ_Pos)  /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos  1          /*!< SCB AIRCR: VECTCLRACTIVE
Position */

#define SCB_AIRCR_VECTCLRACTIVE_Msk  (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos) /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

#define SCB_AIRCR_VECTRESET_Pos     0          /*!< SCB AIRCR: VECTRESET
Position */

#define SCB_AIRCR_VECTRESET_Msk     (1UL << SCB_AIRCR_VECTRESET_Pos)    /*!< SCB
AIRCR: VECTRESET Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos       4          /*!< SCB SCR: SEVONPEND
Position */

#define SCB_SCR_SEVONPEND_Msk       (1UL << SCB_SCR_SEVONPEND_Pos)    /*!< SCB SCR:
SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos       2          /*!< SCB SCR: SLEEPDEEP Position
*/

#define SCB_SCR_SLEEPDEEP_Msk       (1UL << SCB_SCR_SLEEPDEEP_Pos)    /*!< SCB SCR:
SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos     1          /*!< SCB SCR: SLEEPONEXIT
Position */

```

```

#define SCB_SCR_SLEEPONEXIT_Msk      (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR:
SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos        9                                /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk        (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR:
STKALIGN Mask */

#define SCB_CCR_BFHFMIGN_Pos        8                                /*!< SCB CCR: BFHFMIGN
Position */
#define SCB_CCR_BFHFMIGN_Msk        (1UL << SCB_CCR_BFHFMIGN_Pos)      /*!< SCB CCR:
BFHFMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos        4                                /*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk        (1UL << SCB_CCR_DIV_0_TRP_Pos)      /*!< SCB CCR:
DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos      3                                /*!< SCB CCR: UNALIGN_TRP
Position */
#define SCB_CCR_UNALIGN_TRP_Msk      (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB
CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos     1                                /*!< SCB CCR: USERSETMPEND
Position */
#define SCB_CCR_USERSETMPEND_Msk     (1UL << SCB_CCR_USERSETMPEND_Pos)      /*!< SCB
CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos   0                                /*!< SCB CCR:
NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDENA_Msk   (1UL << SCB_CCR_NONBASETHRDENA_Pos)      /*!<
SCB CCR: NONBASETHRDENA Mask */

```

```

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_USGFAULTENA_Pos      18          /*!< SCB SHCSR:  
USGFAULTENA Position */

#define SCB_SHCSR_USGFAULTENA_Msk     (1UL << SCB_SHCSR_USGFAULTENA_Pos)  /*!< SCB  
SHCSR: USGFAULTENA Mask */

#define SCB_SHCSR_BUSFAULTENA_Pos    17          /*!< SCB SHCSR: BUSFAULTENA  
Position */

#define SCB_SHCSR_BUSFAULTENA_Msk   (1UL << SCB_SHCSR_BUSFAULTENA_Pos)  /*!< SCB  
SHCSR: BUSFAULTENA Mask */

#define SCB_SHCSR_MEMFAULTENA_Pos    16          /*!< SCB SHCSR:  
MEMFAULTENA Position */

#define SCB_SHCSR_MEMFAULTENA_Msk   (1UL << SCB_SHCSR_MEMFAULTENA_Pos)  /*!<  
SCB SHCSR: MEMFAULTENA Mask */

#define SCB_SHCSR_SVCALLPENDED_Pos   15          /*!< SCB SHCSR:  
SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk  (1UL << SCB_SHCSR_SVCALLPENDED_Pos)  /*!< SCB  
SHCSR: SVCALLPENDED Mask */

#define SCB_SHCSR_BUSFAULTPENDED_Pos  14          /*!< SCB SHCSR:  
BUSFAULTPENDED Position */

#define SCB_SHCSR_BUSFAULTPENDED_Msk (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)  /*!<  
SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos  13          /*!< SCB SHCSR:  
MEMFAULTPENDED Position */

#define SCB_SHCSR_MEMFAULTPENDED_Msk (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)  /*!<  
SCB SHCSR: MEMFAULTPENDED Mask */

#define SCB_SHCSR_USGFAULTPENDED_Pos  12          /*!< SCB SHCSR:  
USGFAULTPENDED Position */

```

```
#define SCB_SHCSR_USGFAULTPENDED_Msk      (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)      /*!<  
SCB SHCSR: USGFAULTPENDED Mask */  
  
#define SCB_SHCSR_SYSTICKACT_Pos          11                                /*!< SCB SHCSR: SYSTICKACT  
Position */  
#define SCB_SHCSR_SYSTICKACT_Msk          (1UL << SCB_SHCSR_SYSTICKACT_Pos)      /*!< SCB  
SHCSR: SYSTICKACT Mask */  
  
#define SCB_SHCSR_PENDSVACT_Pos           10                                /*!< SCB SHCSR: PENDSVACT  
Position */  
#define SCB_SHCSR_PENDSVACT_Msk           (1UL << SCB_SHCSR_PENDSVACT_Pos)      /*!< SCB  
SHCSR: PENDSVACT Mask */  
  
#define SCB_SHCSR_MONITORACT_Pos          8                                 /*!< SCB SHCSR: MONITORACT  
Position */  
#define SCB_SHCSR_MONITORACT_Msk          (1UL << SCB_SHCSR_MONITORACT_Pos)      /*!< SCB  
SHCSR: MONITORACT Mask */  
  
#define SCB_SHCSR_SVCALLACT_Pos           7                                 /*!< SCB SHCSR: SVCALLACT  
Position */  
#define SCB_SHCSR_SVCALLACT_Msk           (1UL << SCB_SHCSR_SVCALLACT_Pos)      /*!< SCB  
SHCSR: SVCALLACT Mask */  
  
#define SCB_SHCSR_USGFAULTACT_Pos         3                                 /*!< SCB SHCSR: USGFAULTACT  
Position */  
#define SCB_SHCSR_USGFAULTACT_Msk         (1UL << SCB_SHCSR_USGFAULTACT_Pos)      /*!< SCB  
SHCSR: USGFAULTACT Mask */  
  
#define SCB_SHCSR_BUSFAULTACT_Pos         1                                 /*!< SCB SHCSR: BUSFAULTACT  
Position */  
#define SCB_SHCSR_BUSFAULTACT_Msk         (1UL << SCB_SHCSR_BUSFAULTACT_Pos)      /*!< SCB  
SHCSR: BUSFAULTACT Mask */
```

```

#define SCB_SHCSR_MEMFAULTACT_Pos      0                  /*!< SCB SHCSR:  

MEMFAULTACT Position */  
  

#define SCB_SHCSR_MEMFAULTACT_Msk     (1UL << SCB_SHCSR_MEMFAULTACT_Pos)    /*!<  

SCB SHCSR: MEMFAULTACT Mask */  
  

/* SCB Configurable Fault Status Registers Definitions */  
  

#define SCB_CFSR_USGFAULTSR_Pos      16                 /*!< SCB CFSR: Usage Fault  

Status Register Position */  
  

#define SCB_CFSR_USGFAULTSR_Msk      (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos)   /*!< SCB  

CFSR: Usage Fault Status Register Mask */  
  

#define SCB_CFSR_BUSFAULTSR_Pos      8                   /*!< SCB CFSR: Bus Fault Status  

Register Position */  
  

#define SCB_CFSR_BUSFAULTSR_Msk      (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos)    /*!< SCB  

CFSR: Bus Fault Status Register Mask */  
  

#define SCB_CFSR_MEMFAULTSR_Pos      0                  /*!< SCB CFSR: Memory Manage  

Fault Status Register Position */  
  

#define SCB_CFSR_MEMFAULTSR_Msk      (0xFFUL << SCB_CFSR_MEMFAULTSR_Pos)    /*!< SCB  

CFSR: Memory Manage Fault Status Register Mask */  
  

/* SCB Hard Fault Status Registers Definitions */  
  

#define SCB_HFSR_DEBUGEVT_Pos        31                 /*!< SCB HFSR: DEBUGEVT  

Position */  
  

#define SCB_HFSR_DEBUGEVT_Msk        (1UL << SCB_HFSR_DEBUGEVT_Pos)       /*!< SCB HFSR:  

DEBUGEVT Mask */  
  

#define SCB_HFSR_FORCED_Pos         30                 /*!< SCB HFSR: FORCED Position */  
  

#define SCB_HFSR_FORCED_Msk         (1UL << SCB_HFSR_FORCED_Pos)       /*!< SCB HFSR:  

FORCED Mask */  
  

#define SCB_HFSR_VECTTBL_Pos        1                  /*!< SCB HFSR: VECTTBL Position */

```

```

#define SCB_HFSR_VECTTBL_Msk      (1UL << SCB_HFSR_VECTTBL_Pos)      /*!< SCB HFSR:
VECTTBL Mask */

/* SCB Debug Fault Status Register Definitions */

#define SCB_DFSR_EXTERNAL_Pos     4                                /*!< SCB DFSR: EXTERNAL Position
*/
#define SCB_DFSR_EXTERNAL_Msk    (1UL << SCB_DFSR_EXTERNAL_Pos)      /*!< SCB DFSR:
EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos       3                                /*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk     (1UL << SCB_DFSR_VCATCH_Pos)      /*!< SCB DFSR:
VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos      2                                /*!< SCB DFSR: DWTTRAP Position
*/
#define SCB_DFSR_DWTTRAP_Msk    (1UL << SCB_DFSR_DWTTRAP_Pos)      /*!< SCB DFSR:
DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos         1                                /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk       (1UL << SCB_DFSR_BKPT_Pos)      /*!< SCB DFSR: BKPT
Mask */

#define SCB_DFSR_HALTED_Pos       0                                /*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk     (1UL << SCB_DFSR_HALTED_Pos)      /*!< SCB DFSR:
HALTED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
\brief Type definitions for the System Control and ID Register not in the SCB

```

```

@{
}

/** \brief Structure type to access the System Control and ID Register not in the SCB.
 */
typedef struct
{
    uint32_t RESERVED0[1];

    __I uint32_t ICTR;           /*!< Offset: 0x004 (R/ ) Interrupt Controller Type Register */
    __IO uint32_t ACTLR;         /*!< Offset: 0x008 (R/W) Auxiliary Control Register */
} SCnSCB_Type;

/* Interrupt Controller Type Register Definitions */

#define SCnSCB_ICTR_INTLINESNUM_Pos     0           /*!< ICTR: INTLINESNUM Position */
#define SCnSCB_ICTR_INTLINESNUM_Msk      (0xFUL << SCnSCB_ICTR_INTLINESNUM_Pos)  /*!< ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define SCnSCB_ACTLR_DISOOFP_Pos       9           /*!< ACTLR: DISOOFP Position */
#define SCnSCB_ACTLR_DISOOFP_Msk        (1UL << SCnSCB_ACTLR_DISOOFP_Pos)  /*!< ACTLR: DISOOFP Mask */

#define SCnSCB_ACTLR_DISFPCA_Pos       8           /*!< ACTLR: DISFPCA Position */
#define SCnSCB_ACTLR_DISFPCA_Msk        (1UL << SCnSCB_ACTLR_DISFPCA_Pos)  /*!< ACTLR: DISFPCA Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos       2           /*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk        (1UL << SCnSCB_ACTLR_DISFOLD_Pos)  /*!< ACTLR: DISFOLD Mask */

```

```

#define SCnSCB_ACTLR_DISDEFWBUF_Pos      1           /*!< ACTLR: DISDEFWBUF
Position */

#define SCnSCB_ACTLR_DISDEFWBUF_Msk     (1UL << SCnSCB_ACTLR_DISDEFWBUF_Pos)    /*!<
ACTLR: DISDEFWBUF Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos    0           /*!< ACTLR: DISMCYCINT
Position */

#define SCnSCB_ACTLR_DISMCYCINT_Msk   (1UL << SCnSCB_ACTLR_DISMCYCINT_Pos)    /*!<
ACTLR: DISMCYCINT Mask */

/*@} end of group CMSIS_SCnotSCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SysTick System Tick Timer (SysTick)
\brief Type definitions for the System Timer Registers.

@{

*/

```

/** \brief Structure type to access the System Timer (SysTick).

```

*/

```

typedef struct

```

{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;         /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;          /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I  uint32_t CALIB;        /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
}
```

} SysTick_Type;

```

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos      16          /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk     (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2           /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk    (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos      1           /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk     (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos       0           /*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk     (1UL << SysTick_CTRL_ENABLE_Pos)      /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos      0           /*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFUL << SysTick_LOAD_RELOAD_Pos)      /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos      0           /*!< SysTick VAL: CURRENT Position */
#define SysTick_VAL_CURRENT_Msk     (0xFFFFFUL << SysTick_VAL_CURRENT_Pos)      /*!< SysTick VAL: CURRENT Mask */

```

```

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos      31          /*!< SysTick CALIB: NOREF Position */
/* */

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)      /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos      30          /*!< SysTick CALIB: SKEW Position */
/* */

#define SysTick_CALIB_SKEW_Msk     (1UL << SysTick_CALIB_SKEW_Pos)      /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos      0           /*!< SysTick CALIB: TENMS Position */
/* */

#define SysTick_CALIB_TENMS_Msk    (0xFFFFFUL << SysTick_VAL_CURRENT_Pos)  /*!< SysTick
CALIB: TENMS Mask */

/*}@} end of group CMSIS_SysTick */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_ITM  Instrumentation Trace Macrocell (ITM)
\brief Type definitions for the Instrumentation Trace Macrocell (ITM)

@{
*/

```

```
/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
*/

```

```

typedef struct
{
    __O union
{

```

```

__O uint8_t u8;           /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit      */
__O uint16_t u16;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit     */
__O uint32_t u32;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit     */
} PORT [32];              /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers   */

uint32_t RESERVED0[864];

__IO uint32_t TER;         /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register    */
uint32_t RESERVED1[15];

__IO uint32_t TPR;         /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register */
uint32_t RESERVED2[15];

__IO uint32_t TCR;         /*!< Offset: 0xE80 (R/W) ITM Trace Control Register   */
uint32_t RESERVED3[29];

__O uint32_t IWR;          /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register */
__I uint32_t IRR;          /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register */
__IO uint32_t IMCR;         /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
uint32_t RESERVED4[43];

__O uint32_t LAR;          /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register     */
__I uint32_t LSR;          /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register     */
uint32_t RESERVED5[6];

__I uint32_t PID4;          /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
__I uint32_t PID5;          /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
__I uint32_t PID6;          /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
__I uint32_t PID7;          /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
__I uint32_t PID0;          /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
__I uint32_t PID1;          /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
__I uint32_t PID2;          /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
__I uint32_t PID3;          /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
__I uint32_t CID0;          /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
__I uint32_t CID1;          /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
__I uint32_t CID2;          /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */

```

```

__I uint32_t CID3;           /*!< Offset: 0xFFC (R/ ) ITM Component Identification Register #3 */

} ITM_Type;

/* ITM Trace Privilege Register Definitions */

#define ITM_TPR_PRIVMASK_Pos      0           /*!< ITM TPR: PRIVMASK Position */
#define ITM_TPR_PRIVMASK_Msk      (0xFUL << ITM_TPR_PRIVMASK_Pos)    /*!< ITM TPR: PRIVMASK Mask */

/* ITM Trace Control Register Definitions */

#define ITM_TCR_BUSY_Pos         23          /*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk         (1UL << ITM_TCR_BUSY_Pos)        /*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos   16          /*!< ITM TCR: ATBID Position */
#define ITM_TCR_TraceBusID_Msk   (0x7FUL << ITM_TCR_TraceBusID_Pos)  /*!< ITM TCR: ATBID Mask */

#define ITM_TCR_GTSFREQ_Pos      10          /*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk      (3UL << ITM_TCR_GTSFREQ_Pos)       /*!< ITM TCR: Global timestamp frequency Mask */

#define ITM_TCR_TSPrescale_Pos   8           /*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk   (3UL << ITM_TCR_TSPrescale_Pos)     /*!< ITM TCR: TSPrescale Mask */

#define ITM_TCR_SWOENA_Pos       4            /*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk       (1UL << ITM_TCR_SWOENA_Pos)        /*!< ITM TCR: SWOENA Mask */

#define ITM_TCR_DWTENA_Pos       3            /*!< ITM TCR: DWTENA Position */

```

```

#define ITM_TCR_DWTENA_Msk      (1UL << ITM_TCR_DWTENA_Pos)      /*!< ITM TCR:  

DWTENA Mask */

#define ITM_TCR_SYNCENA_Pos     2                                /*!< ITM TCR: SYNCENA Position */  

#define ITM_TCR_SYNCENA_Msk     (1UL << ITM_TCR_SYNCENA_Pos)      /*!< ITM TCR:  

SYNCENA Mask */

#define ITM_TCR_TSENA_Pos       1                                /*!< ITM TCR: TSENA Position */  

#define ITM_TCR_TSENA_Msk       (1UL << ITM_TCR_TSENA_Pos)      /*!< ITM TCR: TSENA  

Mask */

#define ITM_TCR_ITMENA_Pos      0                                /*!< ITM TCR: ITM Enable bit  

Position */  

#define ITM_TCR_ITMENA_Msk      (1UL << ITM_TCR_ITMENA_Pos)      /*!< ITM TCR: ITM  

Enable bit Mask */

/* ITM Integration Write Register Definitions */  

#define ITM_IWR_ATVALIDDM_Pos   0                                /*!< ITM IWR: ATVALIDDM Position  

*/  

#define ITM_IWR_ATVALIDDM_Msk   (1UL << ITM_IWR_ATVALIDDM_Pos)      /*!< ITM IWR:  

ATVALIDDM Mask */

/* ITM Integration Read Register Definitions */  

#define ITMIRR_ATREADYM_Pos    0                                /*!< ITM IRR: ATREADYM Position  

*/  

#define ITMIRR_ATREADYM_Msk    (1UL << ITMIRR_ATREADYM_Pos)      /*!< ITM IRR:  

ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */  

#define ITM_IMCR_INTEGRATION_Pos 0                                /*!< ITM IMCR: INTEGRATION  

Position */  

#define ITM_IMCR_INTEGRATION_Msk (1UL << ITM_IMCR_INTEGRATION_Pos)      /*!< ITM  

IMCR: INTEGRATION Mask */

```

```

/* ITM Lock Status Register Definitions */

#define ITM_LSR_ByteAcc_Pos      2                  /*!< ITM LSR: ByteAcc Position */
#define ITM_LSR_ByteAcc_Msk      (1UL << ITM_LSR_ByteAcc_Pos)    /*!< ITM LSR: ByteAcc Mask */

#define ITM_LSR_Access_Pos       1                  /*!< ITM LSR: Access Position */
#define ITM_LSR_Access_Msk      (1UL << ITM_LSR_Access_Pos)    /*!< ITM LSR: Access Mask */

#define ITM_LSR_Present_Pos     0                  /*!< ITM LSR: Present Position */
#define ITM_LSR_Present_Msk     (1UL << ITM_LSR_Present_Pos)    /*!< ITM LSR: Present Mask */

/*}@*/ /* end of group CMSIS_ITM */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
 \brief Type definitions for the Data Watchpoint and Trace (DWT)

 @{
 */

```

```
/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).
```

```
*/
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) Control Register */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle Count Register */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI Count Register */
}
```

```

__IO uint32_t EXCCNT;           /*!< Offset: 0x00C (R/W) Exception Overhead Count Register */
__IO uint32_t SLEEP_CNT;        /*!< Offset: 0x010 (R/W) Sleep Count Register */
__IO uint32_t LSUCNT;           /*!< Offset: 0x014 (R/W) LSU Count Register */
__IO uint32_t FOLDCNT;          /*!< Offset: 0x018 (R/W) Folded-instruction Count Register */
__I uint32_t PCSR;              /*!< Offset: 0x01C (R) Program Counter Sample Register */
__IO uint32_t COMPO;             /*!< Offset: 0x020 (R/W) Comparator Register 0 */
__IO uint32_t MASK0;             /*!< Offset: 0x024 (R/W) Mask Register 0 */
__IO uint32_t FUNCTION0;         /*!< Offset: 0x028 (R/W) Function Register 0 */
                               uint32_t RESERVED0[1];
__IO uint32_t COMP1;             /*!< Offset: 0x030 (R/W) Comparator Register 1 */
__IO uint32_t MASK1;             /*!< Offset: 0x034 (R/W) Mask Register 1 */
__IO uint32_t FUNCTION1;         /*!< Offset: 0x038 (R/W) Function Register 1 */
                               uint32_t RESERVED1[1];
__IO uint32_t COMP2;             /*!< Offset: 0x040 (R/W) Comparator Register 2 */
__IO uint32_t MASK2;             /*!< Offset: 0x044 (R/W) Mask Register 2 */
__IO uint32_t FUNCTION2;         /*!< Offset: 0x048 (R/W) Function Register 2 */
                               uint32_t RESERVED2[1];
__IO uint32_t COMP3;             /*!< Offset: 0x050 (R/W) Comparator Register 3 */
__IO uint32_t MASK3;             /*!< Offset: 0x054 (R/W) Mask Register 3 */
__IO uint32_t FUNCTION3;         /*!< Offset: 0x058 (R/W) Function Register 3 */

} DWT_Type;

```

```

/* DWT Control Register Definitions */

#define DWT_CTRL_NUMCOMP_Pos      28           /*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk       (0xFUL << DWT_CTRL_NUMCOMP_Pos)    /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos     27           /*!< DWT CTRL: NOTRCPKT Position */

```

#define DWT_CTRL_NOTRCPKT_Msk	(0x1UL << DWT_CTRL_NOTRCPKT_Pos)	/*!< DWT
CTRL: NOTRCPKT Mask */		
#define DWT_CTRL_NOEXTTRIG_Pos	26	/*!< DWT CTRL: NOEXTTRIG
Position */		
#define DWT_CTRL_NOEXTTRIG_Msk	(0x1UL << DWT_CTRL_NOEXTTRIG_Pos)	/*!< DWT
CTRL: NOEXTTRIG Mask */		
#define DWT_CTRL_NOCYCCNT_Pos	25	/*!< DWT CTRL: NOCYCCNT
Position */		
#define DWT_CTRL_NOCYCCNT_Msk	(0x1UL << DWT_CTRL_NOCYCCNT_Pos)	/*!< DWT
CTRL: NOCYCCNT Mask */		
#define DWT_CTRL_NOPRFCNT_Pos	24	/*!< DWT CTRL: NOPRFCNT
Position */		
#define DWT_CTRL_NOPRFCNT_Msk	(0x1UL << DWT_CTRL_NOPRFCNT_Pos)	/*!< DWT
CTRL: NOPRFCNT Mask */		
#define DWT_CTRL_CYCEVTENA_Pos	22	/*!< DWT CTRL: CYCEVTENA
Position */		
#define DWT_CTRL_CYCEVTENA_Msk	(0x1UL << DWT_CTRL_CYCEVTENA_Pos)	/*!< DWT
CTRL: CYCEVTENA Mask */		
#define DWT_CTRL_FOLDEVTENA_Pos	21	/*!< DWT CTRL: FOLDEVTENA
Position */		
#define DWT_CTRL_FOLDEVTENA_Msk	(0x1UL << DWT_CTRL_FOLDEVTENA_Pos)	/*!< DWT
CTRL: FOLDEVTENA Mask */		
#define DWT_CTRL_LSUEVTENA_Pos	20	/*!< DWT CTRL: LSUEVTENA
Position */		
#define DWT_CTRL_LSUEVTENA_Msk	(0x1UL << DWT_CTRL_LSUEVTENA_Pos)	/*!< DWT
CTRL: LSUEVTENA Mask */		

#define DWT_CTRL_SLEEPEVTENA_Pos	19	/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk	(0x1UL << DWT_CTRL_SLEEPEVTENA_Pos)	/*!< DWT CTRL: SLEEPEVTENA Mask */
#define DWT_CTRL_EXCEVTENA_Pos	18	/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk	(0x1UL << DWT_CTRL_EXCEVTENA_Pos)	/*!< DWT CTRL: EXCEVTENA Mask */
#define DWT_CTRL_CPIEVTEA_Pos	17	/*!< DWT CTRL: CPIEVTEA Position */
#define DWT_CTRL_CPIEVTEA_Msk	(0x1UL << DWT_CTRL_CPIEVTEA_Pos)	/*!< DWT CTRL: CPIEVTEA Mask */
#define DWT_CTRL_EXCTRCENA_Pos	16	/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk	(0x1UL << DWT_CTRL_EXCTRCENA_Pos)	/*!< DWT CTRL: EXCTRCENA Mask */
#define DWT_CTRL_PCSAMPLENA_Pos	12	/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk	(0x1UL << DWT_CTRL_PCSAMPLENA_Pos)	/*!< DWT CTRL: PCSAMPLENA Mask */
#define DWT_CTRL_SYNCTAP_Pos	10	/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk	(0x3UL << DWT_CTRL_SYNCTAP_Pos)	/*!< DWT CTRL: SYNCTAP Mask */
#define DWT_CTRL_CYCTAP_Pos	9	/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk	(0x1UL << DWT_CTRL_CYCTAP_Pos)	/*!< DWT CTRL: CYCTAP Mask */

```

#define DWT_CTRL_POSTINIT_Pos      5          /*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk      (0xFUL << DWT_CTRL_POSTINIT_Pos)    /*!< DWT CTRL:
POSTINIT Mask */

#define DWT_CTRL_POSTPRESET_Pos    1          /*!< DWT CTRL: POSTPRESET
Position */
#define DWT_CTRL_POSTPRESET_Msk    (0xFUL << DWT_CTRL_POSTPRESET_Pos)    /*!< DWT
CTRL: POSTPRESET Mask */

#define DWT_CTRL_CYCCNTENA_Pos     0          /*!< DWT CTRL: CYCCNTENA
Position */
#define DWT_CTRL_CYCCNTENA_Msk    (0x1UL << DWT_CTRL_CYCCNTENA_Pos)    /*!< DWT
CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */

#define DWT_CPICNT_CPICNT_Pos     0          /*!< DWT CPICNT: CPICNT Position */
#define DWT_CPICNT_CPICNT_Msk     (0xFFUL << DWT_CPICNT_CPICNT_Pos)    /*!< DWT
CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */

#define DWT_EXCCNT_EXCCNT_Pos     0          /*!< DWT EXCCNT: EXCCNT
Position */
#define DWT_EXCCNT_EXCCNT_Msk     (0xFFUL << DWT_EXCCNT_EXCCNT_Pos)    /*!< DWT
EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */

#define DWT_SLEEPcnt_SLEEPcnt_Pos 0          /*!< DWT SLEEPcnt: SLEEPcnt
Position */
#define DWT_SLEEPcnt_SLEEPcnt_Msk (0xFFUL << DWT_SLEEPcnt_SLEEPcnt_Pos)  /*!< DWT
SLEEPcnt: SLEEPcnt Mask */

```

```

/* DWT LSU Count Register Definitions */

#define DWT_LSUCNT_LSUCNT_Pos      0           /*!< DWT LSUCNT: LSUCNT Position
*/
#define DWT_LSUCNT_LSUCNT_Msk     (0xFFUL << DWT_LSUCNT_LSUCNT_Pos)    /*!< DWT
LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */

#define DWT_FOLDCNT_FOLDCNT_Pos   0           /*!< DWT FOLDCNT: FOLDCNT
Position */
#define DWT_FOLDCNT_FOLDCNT_Msk   (0xFFUL << DWT_FOLDCNT_FOLDCNT_Pos)    /*!< DWT
FOLDCNT: FOLDCNT Mask */

/* DWT Comparator Mask Register Definitions */

#define DWT_MASK_MASK_Pos         0           /*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk         (0x1FUL << DWT_MASK_MASK_Pos)        /*!< DWT MASK:
MASK Mask */

/* DWT Comparator Function Register Definitions */

#define DWT_FUNCTION_MATCHED_Pos   24          /*!< DWT FUNCTION:
MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk   (0x1UL << DWT_FUNCTION_MATCHED_Pos)    /*!<
DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16          /*!< DWT FUNCTION:
DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL << DWT_FUNCTION_DATAVADDR1_Pos) /*!<
DWT FUNCTION: DATAVADDR1 Mask */

#define DWT_FUNCTION_DATAVADDR0_Pos 12          /*!< DWT FUNCTION:
DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk (0xFUL << DWT_FUNCTION_DATAVADDR0_Pos) /*!<
DWT FUNCTION: DATAVADDR0 Mask */

```

```

#define DWT_FUNCTION_DATAVSIZE_Pos      10           /*!< DWT FUNCTION:  

DATAVSIZE Position */  
  

#define DWT_FUNCTION_DATAVSIZE_Msk     (0x3UL << DWT_FUNCTION_DATAVSIZE_Pos)  /*!<  

DWT FUNCTION: DATAVSIZE Mask */  
  

#define DWT_FUNCTION_LNK1ENA_Pos       9            /*!< DWT FUNCTION: LNK1ENA  

Position */  
  

#define DWT_FUNCTION_LNK1ENA_Msk      (0x1UL << DWT_FUNCTION_LNK1ENA_Pos)   /*!<  

DWT FUNCTION: LNK1ENA Mask */  
  

#define DWT_FUNCTION_DATAVMATCH_Pos    8            /*!< DWT FUNCTION:  

DATAVMATCH Position */  
  

#define DWT_FUNCTION_DATAVMATCH_Msk   (0x1UL << DWT_FUNCTION_DATAVMATCH_Pos)  
/*!< DWT FUNCTION: DATAVMATCH Mask */  
  

#define DWT_FUNCTION_CYCMATCH_Pos     7            /*!< DWT FUNCTION:  

CYCMATCH Position */  
  

#define DWT_FUNCTION_CYCMATCH_Msk    (0x1UL << DWT_FUNCTION_CYCMATCH_Pos)   /*!<  

DWT FUNCTION: CYCMATCH Mask */  
  

#define DWT_FUNCTION_EMITRANGE_Pos    5            /*!< DWT FUNCTION:  

EMITRANGE Position */  
  

#define DWT_FUNCTION_EMITRANGE_Msk   (0x1UL << DWT_FUNCTION_EMITRANGE_Pos)  /*!<  

DWT FUNCTION: EMITRANGE Mask */  
  

#define DWT_FUNCTION_FUNCTION_Pos     0            /*!< DWT FUNCTION:  

FUNCTION Position */  
  

#define DWT_FUNCTION_FUNCTION_Msk    (0xFUL << DWT_FUNCTION_FUNCTION_Pos)  /*!<  

DWT FUNCTION: FUNCTION Mask */  
  

/*@}*/ /* end of group CMSIS_DWT */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_TPI Trace Port Interface (TPI)
 \brief Type definitions for the Trace Port Interface (TPI)

 @{

 */

/** \brief Structure type to access the Trace Port Interface Register (TPI).
 */

typedef struct
{
    __IO uint32_t SSPSR;           /*!< Offset: 0x000 (R/ ) Supported Parallel Port Size Register */
    __IO uint32_t CSPSR;          /*!< Offset: 0x004 (R/W) Current Parallel Port Size Register */
    uint32_t RESERVED0[2];
    __IO uint32_t ACPR;           /*!< Offset: 0x010 (R/W) Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];
    __IO uint32_t SPPR;           /*!< Offset: 0x0F0 (R/W) Selected Pin Protocol Register */
    uint32_t RESERVED2[131];
    __I uint32_t FFSR;            /*!< Offset: 0x300 (R/ ) Formatter and Flush Status Register */
    __IO uint32_t FFCR;           /*!< Offset: 0x304 (R/W) Formatter and Flush Control Register */
    __I uint32_t FSCR;            /*!< Offset: 0x308 (R/ ) Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];
    __I uint32_t TRIGGER;         /*!< Offset: 0xEE8 (R/ ) TRIGGER */
    __I uint32_t FIFO0;           /*!< Offset: 0xEEC (R/ ) Integration ETM Data */
    __I uint32_t ITATBCTR2;       /*!< Offset: 0xEF0 (R/ ) ITATBCTR2 */
    uint32_t RESERVED4[1];
    __I uint32_t ITATBCTR0;       /*!< Offset: 0xEF8 (R/ ) ITATBCTR0 */
    __I uint32_t FIFO1;           /*!< Offset: 0xEFC (R/ ) Integration ITM Data */
    __IO uint32_t ITCTRL;          /*!< Offset: 0xF00 (R/W) Integration Mode Control */
    uint32_t RESERVED5[39];
}

```

```

__IO uint32_t CLAIMSET;           /*!< Offset: 0xFA0 (R/W) Claim tag set */
__IO uint32_t CLAIMCLR;          /*!< Offset: 0xFA4 (R/W) Claim tag clear */
uint32_t RESERVED7[8];

__I uint32_t DEVID;             /*!< Offset: 0xFC8 (R/ ) TPIU_DEVID */
__I uint32_t DEVTYPE;           /*!< Offset: 0xFCC (R/ ) TPIU_DEVTYPE */

} TPI_Type;

/* TPI Asynchronous Clock Prescaler Register Definitions */

#define TPI_ACPR_PRESCALER_Pos      0           /*!< TPI ACPR: PRESCALER Position */
#define TPI_ACPR_PRESCALER_Msk      (0xFFFFUL << TPI_ACPR_PRESCALER_Pos)    /*!< TPI ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */

#define TPI_SPPR_TXMODE_Pos        0           /*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk        (0x3UL << TPI_SPPR_TXMODE_Pos)       /*!< TPI SPPR: TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */

#define TPI_FFSR_FtNonStop_Pos     3           /*!< TPI FFSR: FtNonStop Position */
#define TPI_FFSR_FtNonStop_Msk     (0x1UL << TPI_FFSR_FtNonStop_Pos)   /*!< TPI FFSR: FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos     2           /*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk     (0x1UL << TPI_FFSR_TCPresent_Pos)  /*!< TPI FFSR: TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos     1           /*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk     (0x1UL << TPI_FFSR_FtStopped_Pos)  /*!< TPI FFSR: FtStopped Mask */

```

```

#define TPI_FFSR_FInProg_Pos      0           /*!< TPI FFSR: FInProg Position */
#define TPI_FFSR_FInProg_Msk     (0x1UL << TPI_FFSR_FInProg_Pos)    /*!< TPI FFSR: FInProg
Mask */

/* TPI Formatter and Flush Control Register Definitions */

#define TPI_FFCR_TrigIn_Pos      8           /*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk     (0x1UL << TPI_FFCR_TrigIn_Pos)    /*!< TPI FFCR: TrigIn Mask
*/

#define TPI_FFCR_EnFCont_Pos      1           /*!< TPI FFCR: EnFCont Position */
#define TPI_FFCR_EnFCont_Msk     (0x1UL << TPI_FFCR_EnFCont_Pos)    /*!< TPI FFCR:
EnFCont Mask */

/* TPI TRIGGER Register Definitions */

#define TPI_TRIGGER_TRIGGER_Pos      0           /*!< TPI TRIGGER: TRIGGER Position */
#define TPI_TRIGGER_TRIGGER_Msk   (0x1UL << TPI_TRIGGER_TRIGGER_Pos)    /*!< TPI
TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */

#define TPI_FIFO0_ITM_ATVALID_Pos      29          /*!< TPI FIFO0: ITM_ATVALID
Position */
#define TPI_FIFO0_ITM_ATVALID_Msk   (0x3UL << TPI_FIFO0_ITM_ATVALID_Pos)    /*!< TPI FIFO0:
ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos      27          /*!< TPI FIFO0: ITM_bytecount
Position */
#define TPI_FIFO0_ITM_bytecount_Msk   (0x3UL << TPI_FIFO0_ITM_bytecount_Pos)    /*!< TPI
FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos      26          /*!< TPI FIFO0: ETM_ATVALID
Position */

```

```

#define TPI_FIFO0_ETM_ATVALID_Msk      (0x3UL << TPI_FIFO0_ETM_ATVALID_Pos)    /*!< TPI
FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos    24                                /*!< TPI FIFO0: ETM_bytecount
Position */

#define TPI_FIFO0_ETM_bytecount_Msk   (0x3UL << TPI_FIFO0_ETM_bytecount_Pos)  /*!< TPI
FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos           16                                /*!< TPI FIFO0: ETM2 Position */

#define TPI_FIFO0_ETM2_Msk          (0xFFUL << TPI_FIFO0_ETM2_Pos)        /*!< TPI FIFO0: ETM2
Mask */

#define TPI_FIFO0_ETM1_Pos           8       /*!< TPI FIFO0: ETM1 Position */

#define TPI_FIFO0_ETM1_Msk          (0xFFUL << TPI_FIFO0_ETM1_Pos)        /*!< TPI FIFO0: ETM1
Mask */

#define TPI_FIFO0_ETM0_Pos           0       /*!< TPI FIFO0: ETM0 Position */

#define TPI_FIFO0_ETM0_Msk          (0xFFUL << TPI_FIFO0_ETM0_Pos)        /*!< TPI FIFO0: ETM0
Mask */

/* TPI ITATBCTR2 Register Definitions */

#define TPI_ITATBCTR2_ATREADY_Pos    0       /*!< TPI ITATBCTR2: ATREADY
Position */

#define TPI_ITATBCTR2_ATREADY_Msk   (0x1UL << TPI_ITATBCTR2_ATREADY_Pos)  /*!< TPI
ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */

#define TPI_FIFO1_ITM_ATVALID_Pos    29     /*!< TPI FIFO1: ITM_ATVALID
Position */

#define TPI_FIFO1_ITM_ATVALID_Msk   (0x3UL << TPI_FIFO1_ITM_ATVALID_Pos)  /*!< TPI FIFO1:
ITM_ATVALID Mask */

```

```

#define TPI_FIFO1_ITM_bytecount_Pos    27          /*!< TPI FIFO1: ITM_bytecount
Position */

#define TPI_FIFO1_ITM_bytecount_Msk   (0x3UL << TPI_FIFO1_ITM_bytecount_Pos) /*!< TPI
FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos    26          /*!< TPI FIFO1: ETM_ATVALID
Position */

#define TPI_FIFO1_ETM_ATVALID_Msk   (0x3UL << TPI_FIFO1_ETM_ATVALID_Pos) /*!< TPI
FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos    24          /*!< TPI FIFO1: ETM_bytecount
Position */

#define TPI_FIFO1_ETM_bytecount_Msk   (0x3UL << TPI_FIFO1_ETM_bytecount_Pos) /*!< TPI
FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos           16          /*!< TPI FIFO1: ITM2 Position */

#define TPI_FIFO1_ITM2_Msk           (0xFFUL << TPI_FIFO1_ITM2_Pos) /*!< TPI FIFO1: ITM2
Mask */

#define TPI_FIFO1_ITM1_Pos           8           /*!< TPI FIFO1: ITM1 Position */

#define TPI_FIFO1_ITM1_Msk           (0xFFUL << TPI_FIFO1_ITM1_Pos) /*!< TPI FIFO1: ITM1
Mask */

#define TPI_FIFO1_ITM0_Pos           0           /*!< TPI FIFO1: ITM0 Position */

#define TPI_FIFO1_ITM0_Msk           (0xFFUL << TPI_FIFO1_ITM0_Pos) /*!< TPI FIFO1: ITM0
Mask */

/* TPI ITATBCTRO Register Definitions */

#define TPI_ITATBCTRO_ATREADY_Pos    0          /*!< TPI ITATBCTRO: ATREADY
Position */

#define TPI_ITATBCTRO_ATREADY_Msk   (0x1UL << TPI_ITATBCTRO_ATREADY_Pos) /*!< TPI
ITATBCTRO: ATREADY Mask */

```

```

/* TPI Integration Mode Control Register Definitions */

#define TPI_ITCTRL_Mode_Pos      0           /*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk     (0x1UL << TPI_ITCTRL_Mode_Pos)    /*!< TPI ITCTRL:
Mode Mask */

/* TPI DEVID Register Definitions */

#define TPI_DEVID_NRZVALID_Pos   11          /*!< TPI DEVID: NRZVALID Position */
#define TPI_DEVID_NRZVALID_Msk   (0x1UL << TPI_DEVID_NRZVALID_Pos)    /*!< TPI DEVID:
NRZVALID Mask */

#define TPI_DEVID_MANCVALID_Pos  10          /*!< TPI DEVID: MANCVALID
Position */
#define TPI_DEVID_MANCVALID_Msk  (0x1UL << TPI_DEVID_MANCVALID_Pos)    /*!< TPI
DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALID_Pos   9           /*!< TPI DEVID: PTINVALID Position */
#define TPI_DEVID_PTINVALID_Msk   (0x1UL << TPI_DEVID_PTINVALID_Pos)    /*!< TPI DEVID:
PTINVALID Mask */

#define TPI_DEVID_MinBufSz_Pos    6           /*!< TPI DEVID: MinBufSz Position */
#define TPI_DEVID_MinBufSz_Msk   (0x7UL << TPI_DEVID_MinBufSz_Pos)    /*!< TPI DEVID:
MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos    5           /*!< TPI DEVID: AsynClkIn Position */
#define TPI_DEVID_AsynClkIn_Msk   (0x1UL << TPI_DEVID_AsynClkIn_Pos)    /*!< TPI DEVID:
AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos  0           /*!< TPI DEVID: NrTraceInput
Position */
#define TPI_DEVID_NrTraceInput_Msk (0xFUL << TPI_DEVID_NrTraceInput_Pos)    /*!< TPI DEVID:
NrTraceInput Mask */

```

```

/* TPI DEVTYPE Register Definitions */

#define TPI_DEVTYPE_SubType_Pos      0           /*!< TPI DEVTYPE: SubType Position
*/
#define TPI_DEVTYPE_SubType_Msk     (0xFUL << TPI_DEVTYPE_SubType_Pos)    /*!< TPI
DEVTYPE: SubType Mask */

#define TPI_DEVTYPE_MajorType_Pos    4           /*!< TPI DEVTYPE: MajorType
Position */
#define TPI_DEVTYPE_MajorType_Msk   (0xFUL << TPI_DEVTYPE_MajorType_Pos)    /*!< TPI
DEVTYPE: MajorType Mask */

```

```
/*}@*/ /* end of group CMSIS_TPI */
```

```

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUs Memory Protection Unit (MPU)
\brief Type definitions for the Memory Protection Unit (MPU)

@{
*/

```

```

/** \brief Structure type to access the Memory Protection Unit (MPU).
*/
typedef struct
{
    __I uint32_t TYPE;          /*!< Offset: 0x000 (R/ ) MPU Type Register */
    __IO uint32_t CTRL;         /*!< Offset: 0x004 (R/W) MPU Control Register */
    __IO uint32_t RNR;          /*!< Offset: 0x008 (R/W) MPU Region RNRber Register */
    __IO uint32_t RBAR;         /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */

```

```

    __IO uint32_t RASR;           /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register
*/
    __IO uint32_t RBAR_A1;        /*!< Offset: 0x014 (R/W) MPU Alias 1 Region Base Address Register
*/
    __IO uint32_t RASR_A1;        /*!< Offset: 0x018 (R/W) MPU Alias 1 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A2;        /*!< Offset: 0x01C (R/W) MPU Alias 2 Region Base Address Register
*/
    __IO uint32_t RASR_A2;        /*!< Offset: 0x020 (R/W) MPU Alias 2 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A3;        /*!< Offset: 0x024 (R/W) MPU Alias 3 Region Base Address Register
*/
    __IO uint32_t RASR_A3;        /*!< Offset: 0x028 (R/W) MPU Alias 3 Region Attribute and Size
Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)      /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)      /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL << MPU_TYPE_SEPARATE_Pos)      /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

```

```

#define MPU_CTRL_PRIVDEFENA_Pos      2           /*!< MPU CTRL: PRIVDEFENA
Position */

#define MPU_CTRL_PRIVDEFENA_Msk     (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos      1           /*!< MPU CTRL: HFNMIENA
Position */

#define MPU_CTRL_HFNMIENA_Msk     (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos        0           /*!< MPU CTRL: ENABLE Position */

#define MPU_CTRL_ENABLE_Msk       (1UL << MPU_CTRL_ENABLE_Pos)    /*!< MPU CTRL:
ENABLE Mask */

/* MPU Region Number Register */

#define MPU_RNR_REGION_Pos        0           /*!< MPU RNR: REGION Position */

#define MPU_RNR_REGION_Msk       (0xFFUL << MPU_RNR_REGION_Pos)    /*!< MPU RNR:
REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos         5           /*!< MPU RBAR: ADDR Position */

#define MPU_RBAR_ADDR_Msk       (0x7FFFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos        4           /*!< MPU RBAR: VALID Position */

#define MPU_RBAR_VALID_Msk       (1UL << MPU_RBAR_VALID_Pos)    /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos        0           /*!< MPU RBAR: REGION Position */

#define MPU_RBAR_REGION_Msk       (0xFUL << MPU_RBAR_REGION_Pos)    /*!< MPU
RBAR: REGION Mask */

```

```

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos      16          /*!< MPU RASR: MPU Region
Attribute field Position */

#define MPU_RASR_ATTRS_Msk      (0xFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28          /*!< MPU RASR: ATTRS.XN Position
*/
#define MPU_RASR_XN_Msk         (1UL << MPU_RASR_XN_Pos)        /*!< MPU RASR:
ATTRS.XN Mask */

#define MPU_RASR_AP_Pos         24          /*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk         (0x7UL << MPU_RASR_AP_Pos)       /*!< MPU RASR:
ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos        19          /*!< MPU RASR: ATTRS.TEX Position
*/
#define MPU_RASR_TEX_Msk        (0x7UL << MPU_RASR_TEX_Pos)       /*!< MPU RASR:
ATTRS.TEX Mask */

#define MPU_RASR_S_Pos           18          /*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk          (1UL << MPU_RASR_S_Pos)        /*!< MPU RASR: ATTRS.S
Mask */

#define MPU_RASR_C_Pos           17          /*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk          (1UL << MPU_RASR_C_Pos)        /*!< MPU RASR:
ATTRS.C Mask */

#define MPU_RASR_B_Pos           16          /*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk          (1UL << MPU_RASR_B_Pos)        /*!< MPU RASR:
ATTRS.B Mask */

```

```

#define MPU_RASR_SRD_Pos           8          /*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk          (0xFFUL << MPU_RASR_SRD_Pos)      /*!< MPU RASR: Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos         1          /*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk        (0x1FUL << MPU_RASR_SIZE_Pos)      /*!< MPU RASR: Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos       0          /*!< MPU RASR: Region enable bit Position */
#define MPU_RASR_ENABLE_Msk     (1UL << MPU_RASR_ENABLE_Pos)      /*!< MPU RASR: Region enable bit Disable Mask */

/*}@} end of group CMSIS_MPUS*/
#endif

```

```

#if (__FPU_PRESENT == 1)
/** \ingroup CMSIS_core_register
\defgroup CMSIS_FPU Floating Point Unit (FPU)
\brief Type definitions for the Floating Point Unit (FPU)
@{
*/

```

```

/** \brief Structure type to access the Floating Point Unit (FPU).
*/
typedef struct
{

```

```

    uint32_t RESERVED0[1];

    __IO uint32_t FPCCR;           /*!< Offset: 0x004 (R/W) Floating-Point Context Control Register */
}

__IO uint32_t FPCAR;           /*!< Offset: 0x008 (R/W) Floating-Point Context Address Register */
*/

__IO uint32_t FPDSCR;          /*!< Offset: 0x00C (R/W) Floating-Point Default Status Control
Register */

__I uint32_t MVFR0;           /*!< Offset: 0x010 (R/ ) Media and FP Feature Register 0 */
__I uint32_t MVFR1;           /*!< Offset: 0x014 (R/ ) Media and FP Feature Register 1 */

} FPU_Type;

/* Floating-Point Context Control Register */

#define FPU_FPCCR_AS PEN_Pos      31           /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk       (1UL << FPU_FPCCR_AS PEN_Pos)        /*!< FPCCR: ASPEN
bit Mask */

#define FPU_FPCCR_LSPEN_Pos      30           /*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSPEN_Msk       (1UL << FPU_FPCCR_LSPEN_Pos)        /*!< FPCCR: LSPEN
bit Mask */

#define FPU_FPCCR_MONRDY_Pos     8            /*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MONRDY_Msk     (1UL << FPU_FPCCR_MONRDY_Pos)        /*!< FPCCR:
MONRDY bit Mask */

#define FPU_FPCCR_BFRDY_Pos      6            /*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFRDY_Msk       (1UL << FPU_FPCCR_BFRDY_Pos)        /*!< FPCCR: BFRDY
bit Mask */

#define FPU_FPCCR_MMRDY_Pos      5            /*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMRDY_Msk       (1UL << FPU_FPCCR_MMRDY_Pos)        /*!< FPCCR:
MMRDY bit Mask */

```

```

#define FPU_FPCCR_HFRDY_Pos      4          /*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk      (1UL << FPU_FPCCR_HFRDY_Pos)    /*!< FPCCR:
HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos     3          /*!< FPCCR: processor mode bit
Position */
#define FPU_FPCCR_THREAD_Msk     (1UL << FPU_FPCCR_THREAD_Pos)    /*!< FPCCR:
processor mode active bit Mask */

#define FPU_FPCCR_USER_Pos       1          /*!< FPCCR: privilege level bit
Position */
#define FPU_FPCCR_USER_Msk       (1UL << FPU_FPCCR_USER_Pos)    /*!< FPCCR:
privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos     0          /*!< FPCCR: Lazy state preservation
active bit Position */
#define FPU_FPCCR_LSPACT_Msk     (1UL << FPU_FPCCR_LSPACT_Pos)    /*!< FPCCR: Lazy
state preservation active bit Mask */

/* Floating-Point Context Address Register */

#define FPU_FPCAR_ADDRESS_Pos    3          /*!< FPCAR: ADDRESS bit Position */
#define FPU_FPCAR_ADDRESS_Msk    (0x1FFFFFFFUL << FPU_FPCAR_ADDRESS_Pos) /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */

#define FPU_FPDSCR_AHP_Pos       26         /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk       (1UL << FPU_FPDSCR_AHP_Pos)    /*!< FPDSCR: AHP
bit Mask */

#define FPU_FPDSCR_DN_Pos        25         /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk        (1UL << FPU_FPDSCR_DN_Pos)    /*!< FPDSCR: DN bit
Mask */

```

```

#define FPU_FPDSCR_FZ_Pos      24           /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk      (1UL << FPU_FPDSCR_FZ_Pos)    /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos   22           /*!< FPDSCR: RMode bit Position */
#define FPU_FPDSCR_RMode_Msk   (3UL << FPU_FPDSCR_RMode_Pos)  /*!< FPDSCR:
RMode bit Mask */

/* Media and FP Feature Register 0 */

#define FPU_MVFRO_FP_rounding_modes_Pos 28           /*!< MVFRO: FP rounding
modes bits Position */
#define FPU_MVFRO_FP_rounding_modes_Msk (0xFUL << FPU_MVFRO_FP_rounding_modes_Pos)
/*!< MVFRO: FP rounding modes bits Mask */

#define FPU_MVFRO_Short_vectors_Pos 24           /*!< MVFRO: Short vectors bits
Position */
#define FPU_MVFRO_Short_vectors_Msk (0xFUL << FPU_MVFRO_Short_vectors_Pos) /*!<
MVFRO: Short vectors bits Mask */

#define FPU_MVFRO_Square_root_Pos 20           /*!< MVFRO: Square root bits
Position */
#define FPU_MVFRO_Square_root_Msk (0xFUL << FPU_MVFRO_Square_root_Pos) /*!<
MVFRO: Square root bits Mask */

#define FPU_MVFRO_Divide_Pos     16           /*!< MVFRO: Divide bits Position */
#define FPU_MVFRO_Divide_Msk    (0xFUL << FPU_MVFRO_Divide_Pos) /*!< MVFRO:
Divide bits Mask */

#define FPU_MVFRO_FP_excep_trapping_Pos 12           /*!< MVFRO: FP exception
trapping bits Position */

```

```

#define FPU_MVFR0_FP_excep_trapping_Msk  (0xFUL << FPU_MVFR0_FP_excep_trapping_Pos) /*!<
MVFR0: FP exception trapping bits Mask */

#define FPU_MVFR0_Double_precision_Pos    8                                /*!< MVFR0: Double-precision
bits Position */

#define FPU_MVFR0_Double_precision_Msk   (0xFUL << FPU_MVFR0_Double_precision_Pos) /*!<
MVFR0: Double-precision bits Mask */

#define FPU_MVFR0_Single_precision_Pos   4                                /*!< MVFR0: Single-precision
bits Position */

#define FPU_MVFR0_Single_precision_Msk  (0xFUL << FPU_MVFR0_Single_precision_Pos) /*!<
MVFR0: Single-precision bits Mask */

#define FPU_MVFR0_A SIMD_registers_Pos  0                                /*!< MVFR0: A SIMD registers
bits Position */

#define FPU_MVFR0_A SIMD_registers_Msk (0xFUL << FPU_MVFR0_A SIMD_registers_Pos) /*!<
MVFR0: A SIMD registers bits Mask */

/* Media and FP Feature Register 1 */

#define FPU_MVFR1_FP_fused_MAC_Pos     28                               /*!< MVFR1: FP fused MAC
bits Position */

#define FPU_MVFR1_FP_fused_MAC_Msk   (0xFUL << FPU_MVFR1_FP_fused_MAC_Pos) /*!<
MVFR1: FP fused MAC bits Mask */

#define FPU_MVFR1_FP_HPFP_Pos        24                               /*!< MVFR1: FP HPFP bits
Position */

#define FPU_MVFR1_FP_HPFP_Msk       (0xFUL << FPU_MVFR1_FP_HPFP_Pos) /*!< MVFR1:
FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos     4                                /*!< MVFR1: D_NaN mode bits
Position */

#define FPU_MVFR1_D_NaN_mode_Msk   (0xFUL << FPU_MVFR1_D_NaN_mode_Pos) /*!<
MVFR1: D_NaN mode bits Mask */

```

```
#define FPU_MVFR1_FtZ_mode_Pos      0           /*!< MVFR1: FtZ mode bits
Position */

#define FPU_MVFR1_FtZ_mode_Msk      (0xFUL << FPU_MVFR1_FtZ_mode_Pos)    /*!< MVFR1:
FtZ mode bits Mask */

/*}@} end of group CMSIS_FPU */

#endif
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug    Core Debug Registers (CoreDebug)
\brief    Type definitions for the Core Debug Registers
@{
*/

/** \brief Structure type to access the Core Debug Register (CoreDebug).
*/
typedef struct
{
    __IO uint32_t DHCSR;          /*!< Offset: 0x000 (R/W) Debug Halting Control and Status Register */
    __O  uint32_t DCRSR;         /*!< Offset: 0x004 ( /W) Debug Core Register Selector Register */
    __IO uint32_t DCRDR;         /*!< Offset: 0x008 (R/W) Debug Core Register Data Register */
    __IO uint32_t DEMCR;         /*!< Offset: 0x00C (R/W) Debug Exception and Monitor Control
Register */
} CoreDebug_Type;

/* Debug Halting Control and Status Register */
```

```
#define CoreDebug_DHCSR_DBGKEY_Pos    16          /*!< CoreDebug DHCSR:  
DBGKEY Position */  
  
#define CoreDebug_DHCSR_DBGKEY_Msk    (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos)  /*!<  
CoreDebug DHCSR: DBGKEY Mask */  
  
  
#define CoreDebug_DHCSR_S_RESET_ST_Pos 25          /*!< CoreDebug DHCSR:  
S_RESET_ST Position */  
  
#define CoreDebug_DHCSR_S_RESET_ST_Msk (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RESET_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Pos 24          /*!< CoreDebug DHCSR:  
S_RETIRE_ST Position */  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Msk (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RETIRE_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_LOCKUP_Pos   19          /*!< CoreDebug DHCSR:  
S_LOCKUP Position */  
  
#define CoreDebug_DHCSR_S_LOCKUP_Msk  (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)  /*!<  
CoreDebug DHCSR: S_LOCKUP Mask */  
  
  
#define CoreDebug_DHCSR_S_SLEEP_Pos   18          /*!< CoreDebug DHCSR:  
S_SLEEP Position */  
  
#define CoreDebug_DHCSR_S_SLEEP_Msk  (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)  /*!<  
CoreDebug DHCSR: S_SLEEP Mask */  
  
  
#define CoreDebug_DHCSR_S_HALT_Pos    17          /*!< CoreDebug DHCSR:  
S_HALT Position */  
  
#define CoreDebug_DHCSR_S_HALT_Msk  (1UL << CoreDebug_DHCSR_S_HALT_Pos)  /*!<  
CoreDebug DHCSR: S_HALT Mask */  
  
  
#define CoreDebug_DHCSR_S_REGRDY_Pos  16          /*!< CoreDebug DHCSR:  
S_REGRDY Position */  
  
#define CoreDebug_DHCSR_S_REGRDY_Msk (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)  /*!<  
CoreDebug DHCSR: S_REGRDY Mask */
```

```

#define CoreDebug_DHCSR_C_SNAPSTALL_Pos  5          /*!< CoreDebug DHCSR: C_SNAPSTALL Position */

#define CoreDebug_DHCSR_C_SNAPSTALL_Msk  (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */


#define CoreDebug_DHCSR_C_MASKINTS_Pos   3          /*!< CoreDebug DHCSR: C_MASKINTS Position */

#define CoreDebug_DHCSR_C_MASKINTS_Msk  (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)
/*!< CoreDebug DHCSR: C_MASKINTS Mask */


#define CoreDebug_DHCSR_C_STEP_Pos      2          /*!< CoreDebug DHCSR: C_STEP Position */

#define CoreDebug_DHCSR_C_STEP_Msk     (1UL << CoreDebug_DHCSR_C_STEP_Pos)      /*!< CoreDebug DHCSR: C_STEP Mask */


#define CoreDebug_DHCSR_C_HALT_Pos     1          /*!< CoreDebug DHCSR: C_HALT Position */

#define CoreDebug_DHCSR_C_HALT_Msk    (1UL << CoreDebug_DHCSR_C_HALT_Pos)      /*!< CoreDebug DHCSR: C_HALT Mask */


#define CoreDebug_DHCSR_C_DEBUGEN_Pos   0          /*!< CoreDebug DHCSR: C_DEBUGEN Position */

#define CoreDebug_DHCSR_C_DEBUGEN_Msk  (1UL << CoreDebug_DHCSR_C_DEBUGEN_Pos)
/*!< CoreDebug DHCSR: C_DEBUGEN Mask */


/* Debug Core Register Selector Register */

#define CoreDebug_DCRSR_REGWnR_Pos    16         /*!< CoreDebug DCRSR: REGWnR Position */

#define CoreDebug_DCRSR_REGWnR_Msk   (1UL << CoreDebug_DCRSR_REGWnR_Pos)      /*!< CoreDebug DCRSR: REGWnR Mask */

```

```

#define CoreDebug_DCRSR_REGSEL_Pos      0           /*!< CoreDebug DCRSR: REGSEL Position */
#define CoreDebug_DCRSR_REGSEL_Msk     (0x1FUL << CoreDebug_DCRSR_REGSEL_Pos)    /*!< CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos     24          /*!< CoreDebug DEMCR: TRCENA Position */
#define CoreDebug_DEMCR_TRCENA_Msk    (1UL << CoreDebug_DEMCR_TRCENA_Pos)    /*!< CoreDebug DEMCR: TRCENA Mask */

#define CoreDebug_DEMCR_MON_REQ_Pos    19          /*!< CoreDebug DEMCR: MON_REQ Position */
#define CoreDebug_DEMCR_MON_REQ_Msk   (1UL << CoreDebug_DEMCR_MON_REQ_Pos)    /*!< CoreDebug DEMCR: MON_REQ Mask */

#define CoreDebug_DEMCR_MON_STEP_Pos   18          /*!< CoreDebug DEMCR: MON_STEP Position */
#define CoreDebug_DEMCR_MON_STEP_Msk  (1UL << CoreDebug_DEMCR_MON_STEP_Pos)    /*!< CoreDebug DEMCR: MON_STEP Mask */

#define CoreDebug_DEMCR_MON_PEND_Pos   17          /*!< CoreDebug DEMCR: MON_PEND Position */
#define CoreDebug_DEMCR_MON_PEND_Msk  (1UL << CoreDebug_DEMCR_MON_PEND_Pos)    /*!< CoreDebug DEMCR: MON_PEND Mask */

#define CoreDebug_DEMCR_MON_EN_Pos    16          /*!< CoreDebug DEMCR: MON_EN Position */
#define CoreDebug_DEMCR_MON_EN_Msk   (1UL << CoreDebug_DEMCR_MON_EN_Pos)    /*!< CoreDebug DEMCR: MON_EN Mask */

#define CoreDebug_DEMCR_VC_HARDERR_Pos 10          /*!< CoreDebug DEMCR: VC_HARDERR Position */

```

```
#define CoreDebug_DEMCR_VC_HARDERR_Msk  (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos) /*!< CoreDebug DEMCR: VC_HARDERR Mask */

#define CoreDebug_DEMCR_VC_INTERR_Pos    9                         /*!< CoreDebug DEMCR: VC_INTERR Position */

#define CoreDebug_DEMCR_VC_INTERR_Msk   (1UL << CoreDebug_DEMCR_VC_INTERR_Pos) /*!< CoreDebug DEMCR: VC_INTERR Mask */

#define CoreDebug_DEMCR_VC_BUSERR_Pos   8                         /*!< CoreDebug DEMCR: VC_BUSERR Position */

#define CoreDebug_DEMCR_VC_BUSERR_Msk  (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos) /*!< CoreDebug DEMCR: VC_BUSERR Mask */

#define CoreDebug_DEMCR_VC_STATERR_Pos  7                         /*!< CoreDebug DEMCR: VC_STATERR Position */

#define CoreDebug_DEMCR_VC_STATERR_Msk (1UL << CoreDebug_DEMCR_VC_STATERR_Pos) /*!< CoreDebug DEMCR: VC_STATERR Mask */

#define CoreDebug_DEMCR_VC_CHKERR_Pos   6                         /*!< CoreDebug DEMCR: VC_CHKERR Position */

#define CoreDebug_DEMCR_VC_CHKERR_Msk  (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos) /*!< CoreDebug DEMCR: VC_CHKERR Mask */

#define CoreDebug_DEMCR_VC_NOCPERR_Pos  5                         /*!< CoreDebug DEMCR: VC_NOCPERR Position */

#define CoreDebug_DEMCR_VC_NOCPERR_Msk (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos) /*!< CoreDebug DEMCR: VC_NOCPERR Mask */

#define CoreDebug_DEMCR_VC_MMERR_Pos   4                         /*!< CoreDebug DEMCR: VC_MMERR Position */

#define CoreDebug_DEMCR_VC_MMERR_Msk  (1UL << CoreDebug_DEMCR_VC_MMERR_Pos) /*!< CoreDebug DEMCR: VC_MMERR Mask */
```

```

#define CoreDebug_DEMCR_VC_CORERESET_Pos 0          /*!< CoreDebug DEMCR:  

VC_CORERESET Position */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Msk (1UL << CoreDebug_DEMCR_VC_CORERESET_Pos)  
/*!< CoreDebug DEMCR: VC_CORERESET Mask */  
  

/*@} end of group CMSIS_CoreDebug */  
  

  
  

/** \ingroup CMSIS_core_register  

\defgroup CMSIS_core_base Core Definitions  

\brief Definitions for base addresses, unions, and structures.  

@{  

*/  
  

/* Memory mapping of Cortex-M4 Hardware */  
  

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */  

#define ITM_BASE      (0xE0000000UL)           /*!< ITM Base Address */  

#define DWT_BASE      (0xE0001000UL)           /*!< DWT Base Address */  

#define TPI_BASE      (0xE0040000UL)           /*!< TPI Base Address */  

#define CoreDebug_BASE (0xE000EDFOUL)           /*!< Core Debug Base Address */  

#define SysTick_BASE  (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */  

#define NVIC_BASE     (SCS_BASE + 0x0100UL)       /*!< NVIC Base Address */  

#define SCB_BASE      (SCS_BASE + 0x0D00UL)       /*!< System Control Block Base Address */  
  

#define SCnSCB        ((SCnSCB_Type *) SCS_BASE) /*!< System control Register not in SCB */  

#define SCB           ((SCB_Type *) SCB_BASE)      /*!< SCB configuration struct */  

#define SysTick       ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */  

#define NVIC          ((NVIC_Type *) NVIC_BASE)      /*!< NVIC configuration struct */  

#define ITM           ((ITM_Type *) ITM_BASE)        /*!< ITM configuration struct */

```

```

#define DWT      ((DWT_Type *) DWT_BASE) /*!< DWT configuration struct */
#define TPI      ((TPI_Type *) TPI_BASE) /*!< TPI configuration struct */
#define CoreDebug ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration
struct */

#if (__MPU_PRESENT == 1)

#define MPU_BASE   (SCS_BASE + 0x0D90UL) /*!< Memory Protection Unit */
#define MPU       ((MPU_Type *) MPU_BASE) /*!< Memory Protection Unit */

#endif

#if (__FPU_PRESENT == 1)

#define FPU_BASE   (SCS_BASE + 0x0F30UL) /*!< Floating Point Unit */
#define FPU       ((FPU_Type *) FPU_BASE) /*!< Floating Point Unit */

#endif

/*@}*/

```

```

*****  

*      Hardware Abstraction Layer  

Core Function Interface contains:  

- Core NVIC Functions  

- Core SysTick Functions  

- Core Debug Functions  

- Core Register Access Functions  

*****/  

/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference  

*/

```

```

/* ##### NVIC functions #####
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the NVIC.
@{
*/

```

/** \brief Set Priority Grouping

The function sets the priority grouping field using the required unlock sequence.

The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.

Only values from 0..7 are used.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

```

\param [in] PriorityGroup Priority grouping field.
*/
__STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
    uint32_t reg_value;
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07);      /* only values 0..7 are used
*/
    reg_value = SCB->AIRCR;                                         /* read old register configuration */
    reg_value &= ~(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk); /* clear bits to
change          */

```

```

reg_value = (reg_value      |
((uint32_t)0x5FA << SCB_AIRCR_VECTKEY_Pos) |
(PriorityGroupTmp << 8));           /* Insert write key and priority group */
SCB->AIRCR = reg_value;
}

```

/** \brief Get Priority Grouping

The function reads the priority grouping field from the NVIC Interrupt Controller.

```

\return      Priority grouping field (SCB->AIRCR [10:8] PRIGROUP field).
*/
__STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)
{
    return ((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos); /* read priority
grouping field */
}
```

/** \brief Enable External Interrupt

The function enables a device-specific interrupt in the NVIC interrupt controller.

```

\param [in]    IRQn  External interrupt number. Value cannot be negative.
*/
__STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
/* NVIC->ISER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); enable interrupt */

```

```
NVIC->ISER[((uint32_t)((int32_t)IRQn) >> 5) = (uint32_t)(1 << ((uint32_t)((int32_t)IRQn) &  
(uint32_t)0x1F)); /* enable interrupt */  
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]    IRQn  External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* disable interrupt */  
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

```
\param [in]    IRQn  Interrupt number.  
  
\return      0  Interrupt status is not pending.  
\return      1  Interrupt status is pending.  
*/  
STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)  
{
```

```
    return((uint32_t) ((NVIC->ISPR[((uint32_t)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0)); /*  
Return 1 if pending else 0 */  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

```
\param [in] IRQn Interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISPR[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* set interrupt pending */  
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICPR[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* Clear pending interrupt */  
}
```

```
/** \brief Get Active Interrupt
```

The function reads the active register in NVIC and returns the active bit.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not active.

\return 1 Interrupt status is active.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
```

```
{
```

```
    return((uint32_t)((NVIC->IABR[(uint32_t)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0)); /*  
Return 1 if active else 0 */
```

```
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

```
{
```

```
    if(IRQn < 0) {
```

```

SCB->SHP[((uint32_t)(IRQn) & 0xF)-4] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); /* set Priority
for Cortex-M System Interrupts */

else {

    NVIC->IP[(uint32_t)(IRQn)] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); } /* set Priority for
device specific Interrupts */

}

```

/ \brief Get Interrupt Priority**

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

\param [in] IRQn Interrupt number.

\return Interrupt Priority. Value is aligned automatically to the implemented priority bits of the microcontroller.

*/

```

__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)

{

```

if(IRQn < 0) {

```

    return((uint32_t)(SCB->SHP[((uint32_t)(IRQn) & 0xF)-4] >> (8 - __NVIC_PRIO_BITS))); /* get priority
for Cortex-M system interrupts */

```

else {

```

    return((uint32_t)(NVIC->IP[(uint32_t)(IRQn)]      >> (8 - __NVIC_PRIO_BITS))); /* get priority for
device specific interrupts */

```

}

```
/** \brief Encode Priority
```

The function encodes the priority for an interrupt with the given priority group, preemptive priority value, and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

\param [in] PriorityGroup Used priority group.

\param [in] PreemptPriority Preemptive priority value (starting from 0).

\param [in] SubPriority Subpriority value (starting from 0).

\return Encoded priority. Value can be used in the function \ref NVIC_SetPriority().

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority,  
uint32_t SubPriority)
```

```
{
```

```
    uint32_t PriorityGroupTmp = (PriorityGroup & 0x07); /* only values 0..7 are used */
```

```
    uint32_t PreemptPriorityBits;
```

```
    uint32_t SubPriorityBits;
```

```
    PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS : 7 -  
PriorityGroupTmp;
```

```
    SubPriorityBits = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 : PriorityGroupTmp - 7 +  
__NVIC_PRIO_BITS;
```

```
    return (
```

```
        ((PreemptPriority & ((1 << (PreemptPriorityBits)) - 1)) << SubPriorityBits) |
```

```
        ((SubPriority & ((1 << (SubPriorityBits)) - 1)))
```

```
    );
```

```
}
```

```
/** \brief Decode Priority
```

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

\param [in] Priority Priority value, which can be retrieved with the function \ref NVIC_GetPriority().

\param [in] PriorityGroup Used priority group.

\param [out] pPreemptPriority Preemptive priority value (starting from 0).

\param [out] pSubPriority Subpriority value (starting from 0).

*/

```
__STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t*  
pPreemptPriority, uint32_t* pSubPriority)
```

```
{
```

```
    uint32_t PriorityGroupTmp = (PriorityGroup & 0x07); /* only values 0..7 are used */
```

```
    uint32_t PreemptPriorityBits;
```

```
    uint32_t SubPriorityBits;
```

```
    PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS : 7 -  
    PriorityGroupTmp;
```

```
    SubPriorityBits = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 : PriorityGroupTmp - 7 +  
    __NVIC_PRIO_BITS;
```

```
*pPreemptPriority = (Priority >> SubPriorityBits) & ((1 << (PreemptPriorityBits)) - 1);
```

```
*pSubPriority = (Priority & ((1 << (SubPriorityBits)) - 1));
```

```
}
```

```

/** \brief System Reset

The function initiates a system reset request to reset the MCU.

*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                /* Ensure all outstanding memory accesses included
                                              buffered write are completed before reset */

    SCB->AIRCR = ((0x5FA << SCB_AIRCR_VECTKEY_Pos) |

                    (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |

                    SCB_AIRCR_SYSRESETREQ_Msk);           /* Keep priority group unchanged */

    __DSB();                                /* Ensure completion of memory access */

    while(1);                               /* wait until reset */

}

/*@} end of CMSIS_Core_NVICFunctions */

```

```

/* ##### SysTick function
#####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.

 @{
 */

```

```
#if (__Vendor_SysTickConfig == 0)
```

```
/** \brief System Tick Configuration
```

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.

Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.

\return 1 Function failed.

\note When the variable **__Vendor_SysTickConfig** is set to 1, then the function **SysTick_Config** is not included. In this case, the file **<i>device</i>.h** must contain a vendor-specific implementation of this function.

```
*/
```

```
_STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1) > SysTick_LOAD_RELOAD_Msk) return (1); /* Reload value impossible */

    SysTick->LOAD = ticks - 1; /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); /* set Priority for Systick Interrupt */
    SysTick->VAL = 0; /* Load the SysTick Counter Value */
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
        SysTick_CTRL_TICKINT_Msk |
        SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */
    return (0); /* Function successful */
}
```

```
#endif
```

```
/*}@} end of CMSIS_Core_SysTickFunctions */
```

```
/* ##### Debug In/Output function
##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_core_DebugFunctions ITM Functions
 \brief Functions that access the ITM debug interface.

 @{
 */

```

```
extern volatile int32_t ITM_RxBuffer;           /*!< External variable to receive characters.
*/

```

```
#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is
ready for next character. */

```

```
/** \brief ITM Send Character
```

The function transmits a character via the ITM channel 0, and

\li Just returns when no debugger is connected that has booked the output.

\li Is blocking when a debugger is connected, but the previous character sent has not been transmitted.

\param [in] ch Character to transmit.

```

\returns Character to transmit.

*/
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)
{
    if ((ITM->TCR & ITM_TCR_ITMENA_Msk)         && /* ITM enabled */
        (ITM->TER & (1UL << 0)) ) /* ITM Port #0 enabled */
    {
        while (ITM->PORT[0].u32 == 0);
        ITM->PORT[0].u8 = (uint8_t) ch;
    }
    return (ch);
}

```

/** \brief ITM Receive Character

The function inputs a character via the external variable \ref ITM_RxBuffer.

```

\return Received character.

\return -1 No character pending.

*/
__STATIC_INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1; /* no character available */

    if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
        ch = ITM_RxBuffer;
        ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
    }
}
```

```
    return (ch);  
}  
  
}
```

```
/** \brief ITM Check Character
```

The function checks whether a character is pending for reading in the variable `\ref ITM_RxBuffer`.

```
\return 0 No character available.  
\return 1 Character available.  
*/  
STATIC_INLINE int32_t ITM_CheckChar (void) {  
  
    if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {  
        return (0); /* no character available */  
    } else {  
        return (1); /* character available */  
    }  
}  
  
/*@} end of CMSIS_core_DebugFunctions */  
  
#endif /* __CORE_CM4_H_DEPENDANT */  
  
#ifdef __cplusplus  
}  
#endif  
  
#endif /* __CMSIS_GENERIC */
```

```
*****/* @file core_cmFunc.h  
* @brief CMSIS Cortex-M Core Function Access Header File  
* @version V3.30  
* @date 17. February 2014  
*  
* @note  
*  
*****/  
/* Copyright (c) 2009 - 2014 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMFUNC_H
#define __CORE_CMFUNC_H

/* ##### Core Function Access #####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_RegAccFunctions CMSIS Core Register Access Functions
 @{
 */

#ifndef __CC_ARM /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif

/* intrinsic void __enable_irq(); */
/* intrinsic void __disable_irq(); */

/** \brief Get Control Register

```

This function returns the content of the Control Register.

```
\return      Control Register value
*/
__STATIC_INLINE uint32_t __get_CONTROL(void)
{
    register uint32_t __regControl      __ASM("control");
    return(__regControl);
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in]  control  Control Register value to set
*/
__STATIC_INLINE void __set_CONTROL(uint32_t control)
{
    register uint32_t __regControl      __ASM("control");
    __regControl = control;
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return      IPSR Register value
*/
__STATIC_INLINE uint32_t __get_IPSR(void)
{
    register uint32_t __regIPSR      __ASM("ipsr");
    return(__regIPSR);
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return      APSR Register value
*/
__STATIC_INLINE uint32_t __get_APSR(void)
{
    register uint32_t __regAPSR      __ASM("apsr");
    return(__regAPSR);
}
```

```
/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
\return      xPSR Register value
*/
__STATIC_INLINE uint32_t __get_xPSR(void)
```

```
{  
register uint32_t __regXPSR      __ASM("xpsr");  
return(__regXPSR);  
}
```

```
/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
\return      PSP Register value  
*/  
__STATIC_INLINE uint32_t __get_PSP(void)  
{  
register uint32_t __regProcessStackPointer __ASM("psp");  
return(__regProcessStackPointer);  
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in]  topOfProcStack  Process Stack Pointer value to set  
*/  
__STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)  
{  
register uint32_t __regProcessStackPointer __ASM("psp");  
__regProcessStackPointer = topOfProcStack;
```

```
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value
*/
__STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t __regMainStackPointer __ASM("msp");
    return(__regMainStackPointer);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
*/
__STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)
{
    register uint32_t __regMainStackPointer __ASM("msp");
    __regMainStackPointer = topOfMainStack;
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value  
*/  
STATIC_INLINE uint32_t __get_PRIMASK(void)  
{  
    register uint32_t __regPriMask      __ASM("primask");  
    return(__regPriMask);  
}
```

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in]  priMask Priority Mask  
*/  
STATIC_INLINE void __set_PRIMASK(uint32_t priMask)  
{  
    register uint32_t __regPriMask      __ASM("primask");  
    __regPriMask = (priMask);  
}
```

```
#if    (__CORTEX_M >= 0x03)
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

*/

```
#define __enable_fault_irq      __enable_fiq
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

*/

```
#define __disable_fault_irq     __disable_fiq
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

\return Base Priority register value

*/

```
_STATIC_INLINE uint32_t __get_BASEPRI(void)
```

```
{
```

```
    register uint32_t __regBasePri    __ASM("basepri");
```

```
    return(__regBasePri);
```

```
}
```

```
/** \brief Set Base Priority
```

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set  
*/  
STATIC_INLINE void __set_BASEPRI(uint32_t basePri)  
{  
    register uint32_t __regBasePri      __ASM("basepri");  
    __regBasePri = (basePri & 0xff);  
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return      Fault Mask register value  
*/  
STATIC_INLINE uint32_t __get_FAULTMASK(void)  
{  
    register uint32_t __regFaultMask    __ASM("faultmask");  
    return(__regFaultMask);  
}
```

```
/** \brief Set Fault Mask
```

This function assigns the given value to the Fault Mask register.

```
\param [in]  faultMask  Fault Mask value to set
*/
__STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    register uint32_t __regFaultMask      __ASM("faultmask");
    __regFaultMask = (faultMask & (uint32_t)1);
}
```

```
#endif /* (__CORTEX_M >= 0x03) */
```

```
#if      (__CORTEX_M == 0x04)
```

```
/** \brief Get FPSCR
```

This function returns the current value of the Floating Point Status/Control register.

```
\return      Floating Point Status/Control register value
*/
__STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpscr      __ASM("fpscr");
    return(__regfpscr);
#else
    return(0);
#endif
}
```

```
/** \brief Set FPSCR
```

This function assigns the given value to the Floating Point Status/Control register.

```
\param [in]  fpsc  Floating Point Status/Control value to set
*/
__STATIC_INLINE void __set_FPSCR(uint32_t fpsc)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpsc      __ASM("fpsc");
    __regfpsc = (fpsc);
#endif
}
```

```
#endif /* (__CORTEX_M == 0x04) */
```

```
#elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */
```

```
/** \brief Enable IRQ Interrupts
```

This function enables IRQ interrupts by clearing the I-bit in the CPSR.

Can only be executed in Privileged modes.

```
/*
__attribute__( ( always_inline ) ) __STATIC_INLINE void __enable_irq(void)
{
    __ASM volatile ("cpsie i" :: : "memory");
```

```
}
```

```
/** \brief Disable IRQ Interrupts
```

This function disables IRQ interrupts by setting the I-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_irq(void)
{
    __ASM volatile ("cpsid i" : : : "memory");
}
```

```
/** \brief Get Control Register
```

This function returns the content of the Control Register.

```
\return      Control Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_CONTROL(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, control" : "=r" (result));
    return(result);
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in] control Control Register value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_CONTROL(uint32_t control)
{
    __ASM volatile ("MSR control, %0" : : "r" (control) : "memory");
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return      IPSR Register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_IPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, ipsr" : "=r" (result));
    return(result);
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return      APSR Register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_APSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, apsr" : "=r" (result));
    return(result);
}

/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
\return      xPSR Register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_xPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, xpsr" : "=r" (result));
    return(result);
}

/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
\return      PSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_PSP(void)
{
    register uint32_t result;

    __ASM volatile ("MRS %0, psp\n" : "=r" (result) );
    return(result);
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in]  topOfProcStack Process Stack Pointer value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)
{
    __ASM volatile ("MSR psp, %0\n" :: "r" (topOfProcStack) : "sp");
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t result;

    __ASM volatile ("MRS %0, msp\n" : "=r" (result) );
    return(result);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)
{
    __ASM volatile ("MSR msp, %0\n" : : "r" (topOfMainStack) : "sp");
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value
```

```
*/  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_PRIMASK(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, primask" : "=r" (result) );  
    return(result);  
}
```

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in] priMask Priority Mask  
*/  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_PRIMASK(uint32_t priMask)  
{  
    __ASM volatile ("MSR primask, %0" : : "r" (priMask) : "memory");  
}
```

```
#if     (__CORTEX_M >= 0x03)
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE void __enable_fault_irq(void)
{
    __ASM volatile ("cpsie f" : : : "memory");
}
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_fault_irq(void)
{
    __ASM volatile ("cpsid f" : : : "memory");
}
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

```
\return      Base Priority register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_BASEPRI(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, basepri_max" : "=r" (result) );
    return(result);
```

```
}
```

```
/** \brief Set Base Priority
```

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_BASEPRI(uint32_t value)
{
    __ASM volatile ("MSR basepri, %0" : : "r" (value) : "memory");
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return Fault Mask register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_FAULTMASK(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, faultmask" : "=r" (result));
    return(result);
}
```

```

/** \brief Set Fault Mask

This function assigns the given value to the Fault Mask register.

\param [in] faultMask Fault Mask value to set

*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    __ASM volatile ("MSR faultmask, %0" : : "r" (faultMask) : "memory");
}

#endif /* (__CORTEX_M >= 0x03) */

#if     (__CORTEX_M == 0x04)

/** \brief Get FPSCR

This function returns the current value of the Floating Point Status/Control register.

\return      Floating Point Status/Control register value

*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    uint32_t result;

    /* Empty asm statement works as a scheduling barrier */

```

```

__ASM volatile ("");
__ASM volatile ("VMRS %0, fpSCR" : "=r" (result) );
__ASM volatile ("");
return(result);

#else
    return(0);
#endif
}

```

/** \brief Set FPSCR

This function assigns the given value to the Floating Point Status/Control register.

```

\param [in]  fpSCR Floating Point Status/Control value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
/* Empty asm statement works as a scheduling barrier */
__ASM volatile ("");
__ASM volatile ("VMSR fpSCR, %0" : : "r" (fpSCR) : "vfpcc");
__ASM volatile ("");
#endif
}

#endif /* (__CORTEX_M == 0x04) */

```

```
#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

# elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

# elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
 */
# endif

# elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*}@} end of CMSIS_Core_RegAccFunctions */

#endif /* __CORE_CMFUNC_H */
/*********************//**
```

```
* @file core_cm4_simd.h
* @brief CMSIS Cortex-M4 SIMD Header File
* @version V3.30
* @date 17. February 2014
*
* @note
*
*****
/* Copyright (c) 2009 - 2014 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

----- */

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif

#ifndef __CORE_CM4 SIMD_H
#define __CORE_CM4 SIMD_H

#ifdef __cplusplus
extern "C" {

#endif

/*********************  

 *      Hardware Abstraction Layer  

/*******************/
```

/* ##### Compiler specific Intrinsics ##### */
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
Access to dedicated SIMD instructions
@{
*/

```
#if defined (__CC_ARM ) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#define __SADD8           __sadd8
#define __QADD8           __qadd8
#define __SHADD8          __shadd8
#define __UADD8           __uadd8
#define __UQADD8          __uqadd8
#define __UHADD8          __uhadd8
#define __SSUB8            __ssub8
#define __QSUB8            __qsub8
#define __SHSUB8           __shsub8
#define __USUB8            __usub8
#define __UQSUB8           __uqsub8
#define __UHSUB8           __uhsub8
#define __SADD16           __sadd16
#define __QADD16           __qadd16
#define __SHADD16          __shadd16
#define __UADD16           __uadd16
#define __UQADD16          __uqadd16
#define __UHADD16          __uhadd16
#define __SSUB16           __ssub16
#define __QSUB16           __qsub16
#define __SHSUB16          __shsub16
#define __USUB16           __usub16
#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX              __sasx
#define __QASX              __qasx
```

```
#define __SHASX          __shasx
#define __UASX           __uasx
#define __UQASX          __uqasx
#define __UHASX          __uhasl
#define __SSAX            __ssax
#define __QSAX            __qsax
#define __SHSAX          __shsax
#define __USAX            __usax
#define __UQSAX          __uqsax
#define __UHSAX          __uhsax
#define __USAD8           __usad8
#define __USADA8          __usada8
#define __SSAT16          __ssat16
#define __USAT16          __usat16
#define __UXTB16          __uxtb16
#define __UXTAB16         __uxtab16
#define __SXTB16          __sxtb16
#define __SXTAB16         __sxtab16
#define __SMUAD           __smuad
#define __SMUADX          __smuadx
#define __SMLAD           __smlad
#define __SMLADX          __smladx
#define __SMLALD          __smlald
#define __SMLALDX         __smlalidx
#define __SMUSD            __smusd
#define __SMUSDX          __smusdx
#define __SMLSD            __smlsd
#define __SMLSDX          __smlsdx
#define __SMLSLD          __smlsld
```

```

#define __SMLSLDX      __smIsIdx
#define __SEL          __sel
#define __QADD         __qadd
#define __QSUB         __qsub

#define __PKHBT(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0x0000FFFFUL) | \
                                (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0xFFFF0000UL) | \
                                (((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL) )

#define __SMMLA(ARG1,ARG2,ARG3)    ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                ((int64_t)(ARG3) << 32) ) >> 32))

#elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

```

```
__ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD8(uint32_t op1, uint32_t op2)
```



```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD16(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UQSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSAX(uint32_t op1, uint32_t op2)
{
```

```
uint32_t result;

__ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __USAD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __USADA8(uint32_t op1, uint32_t op2,
    uint32_t op3)
{
    uint32_t result;
```

```
__ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
return(result);
}
```

```
#define __SSAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1) ); \
    __RES; \
})
```

```
#define __USAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1) ); \
    __RES; \
})
```

```
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}
```

```
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTAB16(uint32_t op1, uint32_t op2)
{
```

```
uint32_t result;

__ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUAD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUADX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLAD (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLADX (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLALD (uint32_t op1, uint32_t op2,
uint64_t acc)
{

```

```

union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian

__ASM volatile ("smalld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else // Big endian

__ASM volatile ("smalld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian

__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else // Big endian

```

```

__ASM volatile ("smalidx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUSD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUSDX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
}

```

```

return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSLD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2), "0"
(llr.w32[0]), "1" (llr.w32[1]));
#else // Big endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2), "0"
(llr.w32[1]), "1" (llr.w32[0]));
#endif

    return(llr.w64);
}

```

```

}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLSIDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smlsidx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2), "0"
(llr.w32[0]), "1" (llr.w32[1]));
#else // Big endian
__ASM volatile ("smlsidx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2), "0"
(llr.w32[1]), "1" (llr.w32[0]));
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SEL (uint32_t op1, uint32_t op2)
{
    uint32_t result;

__ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

```

```

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QADD(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QSUB(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```

#define __PKHBT(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
    __RES; \
})

```

```

#define __PKHTB(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2) ); \
    else \

```

```

__ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
__RES; \
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMMLA (int32_t op1, int32_t op2, int32_t
op3)
{
    int32_t result;

    __ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING cccm specific functions */
/* not yet supported */

```

```

#ifndef __COSMIC__
#define __COSMIC__ 1
#endif /* __COSMIC__ */

/*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif /* __COSMIC__ */

/*@} end of group CMSIS SIMD intrinsics */

#ifndef __cplusplus
}
#endif /* __cplusplus */

#endif /* __CORE_CM4 SIMD_H */

/*
 * @file core_cmInstr.h
 * @brief CMSIS Cortex-M Core Instruction Access Header File
 * @version V3.30
 * @date 17. February 2014
 *
 * @note
 *
 */
/* Copyright (c) 2009 - 2014 ARM LIMITED
 */

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMINSTR_H  
#define __CORE_CMINSTR_H
```

```
/* ##### Core Instruction Access ##### */  
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface  
Access to dedicated instructions
```

```
@{  
*/  
  
#if defined (__CC_ARM ) /*-----RealView Compiler -----*/  
/* ARM armcc specific functions */
```

```
#if (__ARMCC_VERSION < 400677)  
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"  
#endif
```

```
/** \brief No Operation
```

No Operation does nothing. This instruction can be used for code alignment purposes.

```
*/  
  
#define __NOP          __nop
```

```
/** \brief Wait For Interrupt
```

Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occurs.

```
*/  
  
#define __WFI          __wfi
```

```
/** \brief Wait For Event
```

Wait For Event is a hint instruction that permits the processor to enter

a low-power state until one of a number of events occurs.

*/

```
#define __WFE           __wfe
```

```
/** \brief Send Event
```

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

*/

```
#define __SEV           __sev
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor,
so that all instructions following the ISB are fetched from cache or
memory, after the instruction has been completed.

*/

```
#define __ISB()          __isb(0xF)
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

*/

```
#define __DSB()          __dsb(0xF)
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before and after the instruction, without ensuring their completion.

```
*/
```

```
#define __DMB()           __dmb(0xF)
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

```
\param [in]  value Value to reverse
```

```
\return      Reversed value
```

```
*/
```

```
#define __REV          __rev
```

```
/** \brief Reverse byte order (16 bit)
```

This function reverses the byte order in two unsigned short values.

```
\param [in]  value Value to reverse
```

```
\return      Reversed value
```

```
*/
```

```
#ifndef __NO_EMBEDDED_ASM
```

```
__attribute__((section(".rev16_text"))) __STATIC_INLINE __ASM uint32_t __REV16(uint32_t value)
```

```
{
```

```
    rev16 r0, r0
```

```
    bx lr
}

#endif
```

```
/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".revsh_text"))) __STATIC_INLINE __ASM int32_t __REVSH(int32_t value)
{
    revsh r0, r0
    bx lr
}
#endif
```

```
/** \brief Rotate Right in unsigned value (32 bit)
```

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```
\param [in]  value Value to rotate
\param [in]  value Number of Bits to rotate
\return      Rotated value
*/
```

```
#define __ROR           __ror
```

```
/** \brief Breakpoint
```

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

\param [in] value is ignored by the processor.

If required, a debugger can use it to store additional information about the breakpoint.

```
*/
```

```
#define __BKPT(value)      __breakpoint(value)
```

```
#if    (__CORTEX_M >= 0x03)
```

```
/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

\param [in] value Value to reverse

\return Reversed value

```
*/
```

```
#define __RBIT          __rbit
```

```
/** \brief LDR Exclusive (8 bit)
```

This function performs a exclusive LDR command for 8 bit value.

```
\param [in]  ptr  Pointer to data
\return      value of type uint8_t at (*ptr)
*/
#define __LDREXB(ptr)          ((uint8_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (16 bit)
```

This function performs a exclusive LDR command for 16 bit values.

```
\param [in]  ptr  Pointer to data
\return      value of type uint16_t at (*ptr)
*/
#define __LDREXH(ptr)          ((uint16_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (32 bit)
```

This function performs a exclusive LDR command for 32 bit values.

```
\param [in]  ptr  Pointer to data
\return      value of type uint32_t at (*ptr)
*/
#define __LDREXW(ptr)          ((uint32_t) __ldrex(ptr))
```

```
/** \brief STR Exclusive (8 bit)
```

This function performs a exclusive STR command for 8 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
\return 0 Function succeeded  
\return 1 Function failed  
*/  
#define __STREXB(value, ptr)      __strex(value, ptr)
```

```
/** \brief STR Exclusive (16 bit)
```

This function performs a exclusive STR command for 16 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
\return 0 Function succeeded  
\return 1 Function failed  
*/  
#define __STREXB(value, ptr)      __strex(value, ptr)
```

```
/** \brief STR Exclusive (32 bit)
```

This function performs a exclusive STR command for 32 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location
```

```
\return 0 Function succeeded  
\return 1 Function failed  
*/  
  
#define __STREXW(value, ptr)      __strex(value, ptr)
```

/** \brief Remove the exclusive lock

This function removes the exclusive lock which is created by LDREX.

```
*/  
  
#define __CLREX      __clrex
```

/** \brief Signed Saturate

This function saturates a signed value.

```
\param [in] value Value to be saturated  
\param [in] sat Bit position to saturate to (1..32)  
\return Saturated value  
*/  
  
#define __SSAT      __ssat
```

/** \brief Unsigned Saturate

This function saturates an unsigned value.

```
\param [in] value Value to be saturated  
\param [in] sat Bit position to saturate to (0..31)  
\return Saturated value  
*/  
  
#define __USAT __usat
```

```
/** \brief Count leading zeros
```

This function counts the number of leading zeros of a data value.

```
\param [in] value Value to count the leading zeros  
\return number of leading zeros in value  
*/  
  
#define __CLZ __clz
```

```
#endif /* (__CORTEX_M >= 0x03) */
```

```
#elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/  
/* GNU gcc specific functions */
```

```
/* Define macros for porting to both thumb1 and thumb2.  
 * For thumb1, use low register (r0-r7), specified by constraint "l"  
 * Otherwise, use general registers, specified by constraint "r" */  
  
#if defined (__thumb__) && !defined (__thumb2__)  
  
#define __CMSIS_GCC_OUT_REG(r) "=l" (r)  
#define __CMSIS_GCC_USE_REG(r) "l" (r)  
  
#else
```

```
#define __CMSIS_GCC_OUT_REG(r) "=r" (r)
#define __CMSIS_GCC_USE_REG(r) "r" (r)
#endif

/** \brief No Operation

No Operation does nothing. This instruction can be used for code alignment purposes.

*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __NOP(void)
{
    __ASM volatile ("nop");
}
```

```
/** \brief Wait For Interrupt

Wait For Interrupt is a hint instruction that suspends execution
until one of a number of events occurs.

*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __WFI(void)
{
    __ASM volatile ("wfi");
}
```

```
/** \brief Wait For Event

Wait For Event is a hint instruction that permits the processor to enter
a low-power state until one of a number of events occurs.
```

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __WFE(void)  
{  
    __ASM volatile ("wfe");  
}
```

```
/** \brief Send Event
```

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __SEV(void)  
{  
    __ASM volatile ("sev");  
}
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the instruction has been completed.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __ISB(void)  
{  
    __ASM volatile ("isb");  
}
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

```
*/
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE void __DSB(void)
{
    __ASM volatile ("dsb");
}
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before and after the instruction, without ensuring their completion.

```
*/
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE void __DMB(void)
{
    __ASM volatile ("dmb");
}
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

\param [in] value Value to reverse

\return Reversed value

```
*/
```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __REV(uint32_t value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 5)
    return __builtin_bswap32(value);
#else
    uint32_t result;

    __ASM volatile ("rev %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
#endif
}

```

/** \brief Reverse byte order (16 bit)

This function reverses the byte order in two unsigned short values.

```

\param [in]  value  Value to reverse
\return      Reversed value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __REV16(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("rev16 %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
}

```

```
/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in]  value Value to reverse
\return      Reversed value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE int32_t __REVSH(int32_t value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    return (short)__builtin_bswap16(value);
#else
    uint32_t result;

    __ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
#endif
}

/** \brief Rotate Right in unsigned value (32 bit)
```

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```
\param [in]  value Value to rotate
\param [in]  value Number of Bits to rotate
\return      Rotated value
*/
```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __ROR(uint32_t op1, uint32_t op2)
{
    return (op1 >> op2) | (op1 << (32 - op2));
}
```

```
/** \brief Breakpoint
```

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

\param [in] value is ignored by the processor.

If required, a debugger can use it to store additional information about the breakpoint.

```
*/
#define __BKPT(value)           __ASM volatile ("bkpt "#value)
```

```
#if    (__CORTEX_M >= 0x03)
```

```
/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

```
\param [in] value Value to reverse
\return      Reversed value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __RBIT(uint32_t value)
{
```

```

    uint32_t result;

    __ASM volatile ("rbit %0, %1" : "=r" (result) : "r" (value) );
    return(result);
}

```

/** \brief LDR Exclusive (8 bit)

This function performs a exclusive LDR command for 8 bit value.

```

\param [in]  ptr  Pointer to data
\return     value of type uint8_t at (*ptr)
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint8_t __LDREXB(volatile uint8_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexb %0, %1" : "=r" (result) : "Q" (*addr) );
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.
*/
    __ASM volatile ("ldrexb %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );
#endif
    return ((uint8_t) result); /* Add explicit type cast here */
}

```

```
/** \brief LDR Exclusive (16 bit)
```

This function performs a exclusive LDR command for 16 bit values.

```
\param [in]  ptr  Pointer to data
\return     value of type uint16_t at (*ptr)
*/
__attribute__(( always_inline )) __STATIC_INLINE uint16_t __LDREXH(volatile uint16_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexh %0, %1" : "=r" (result) : "Q" (*addr));
#else
    /* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
       accepted by assembler. So has to use following less efficient pattern.
    */
    __ASM volatile ("ldrexh %0, [%1]" : "=r" (result) : "r" (addr) : "memory");
#endif
    return ((uint16_t) result); /* Add explicit type cast here */
}
```

```
/** \brief LDR Exclusive (32 bit)
```

This function performs a exclusive LDR command for 32 bit values.

```
\param [in]  ptr  Pointer to data
```

```

\return value of type uint32_t at (*ptr)
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __LDREXW(volatile uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrex %0, %1" : "=r" (result) : "Q" (*addr));
    return(result);
}

```

/** \brief STR Exclusive (8 bit)

This function performs a exclusive STR command for 8 bit values.

```

\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __STREXB(uint8_t value, volatile uint8_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strexb %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value));
    return(result);
}

```

```

/** \brief STR Exclusive (16 bit)

This function performs a exclusive STR command for 16 bit values.

\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __STREXH(uint16_t value, volatile uint16_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strexh %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value));
    return(result);
}

```

```

/** \brief STR Exclusive (32 bit)

```

This function performs a exclusive STR command for 32 bit values.

```

\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/

```

```
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __STREXW(uint32_t value, volatile uint32_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" (value) );
    return(result);
}
```

/** \brief Remove the exclusive lock

This function removes the exclusive lock which is created by LDREX.

```
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __CLREX(void)
{
    __ASM volatile ("clrex" ::: "memory");
}
```

/** \brief Signed Saturate

This function saturates a signed value.

```
\param [in] value Value to be saturated
\param [in] sat Bit position to saturate to (1..32)
\return Saturated value
*/
```

```
#define __SSAT(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})
```

```
/** \brief Unsigned Saturate
```

This function saturates an unsigned value.

```
\param [in] value Value to be saturated
\param [in] sat Bit position to saturate to (0..31)
\return Saturated value
*/
#define __USAT(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})
```

```
/** \brief Count leading zeros
```

This function counts the number of leading zeros of a data value.

```
\param [in] value Value to count the leading zeros
```

```

    \return      number of leading zeros in value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint8_t __CLZ(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("clz %0, %1" : "=r" (result) : "r" (value));
    return ((uint8_t) result); /* Add explicit type cast here */
}

#endif /* (__CORTEX_M >= 0x03) */

#ifndef __ICCARM__
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#ifndef __TMS470__
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#ifndef __TASKING__
/* TASKING ccm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
 */

```

```

*/



#ifndef __CORE_CM0PLUS_H
#define __CORE_CM0PLUS_H

/* Cosmic specific functions */

#include <cmsis_csm.h>

#endif /* __CORE_CM0PLUS_H */

/* @file core_cm0plus.h
 * @brief CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
 * @version V3.30
 * @date 24. February 2014
 *
 * @note
 *
 */
/* Copyright (c) 2009 - 2014 ARM LIMITED
 */

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA check */
#endif
```

```
#ifndef __CORE_CM0PLUS_H_GENERIC
```

```
#define __CORE_CM0PLUS_H_GENERIC
```

```
#ifdef __cplusplus
```

```
extern "C" {
```

```
#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions
```

```
CMSIS violates the following MISRA-C:2004 rules:
```

```
\li Required Rule 8.5, object/function definition in header file.<br>
```

```
Function definitions in header files are used to allow 'inlining'.
```

```
\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.<br>
```

```
Unions are used for effective representation of core registers.
```

```
\li Advisory Rule 19.7, Function-like macro defined.<br>
```

```
Function-like macros are used to allow more efficient code.
```

```
*/
```

```
*****
```

```
* CMSIS definitions
```

```
***** /
```

```
/** \ingroup Cortex-M0+
```

```
@{
```

```
*/
```

```
/* CMSIS CM0P definitions */
```

```
#define __CM0PLUS_CMSIS_VERSION_MAIN (0x03) /*!< [31:16] CMSIS HAL main version */
```

```
#define __CM0PLUS_CMSIS_VERSION_SUB (0x20) /*!< [15:0] CMSIS HAL sub version */
```

```
#define __CM0PLUS_CMSIS_VERSION ((__CM0PLUS_CMSIS_VERSION_MAIN << 16) | \
```

```

__CM0PLUS_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */

#define __CORTEX_M          (0x00)           /*!< Cortex-M Core */
                                         */

#if defined (__CC_ARM)
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)
#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)
#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

#elif defined (__TMS470__)
#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#elif defined (__TASKING__)
#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

```

```

#define __packed
#define __ASM      _asm          /*!< asm keyword for COSMIC Compiler */
#define __INLINE    inline        /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */
#define __STATIC_INLINE static inline

#endif

/** __FPU_USED indicates whether an FPU is used or not. This core does not support an FPU at all
*/
#define __FPU_USED    0

#if defined ( __CC_ARM )
#if defined __TARGET_FPU_VFP
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __GNUC__ )
#if defined (__VFP_FP__) && !defined(__SOFTFP__)
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __ICCARM__ )
#if defined __ARMVFP__
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

```

```

#ifndef __CMSIS_GENERIC

#elif defined ( __TMS470__ )

#if defined __TI__VFP_SUPPORT____
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __TASKING__ )

#if defined __FPU_VFP__
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __CSMC__ )           /* Cosmic */
#if ( __CSMC__ & 0x400)           // FPU present for parser
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#endif

#include <stdint.h>          /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access        */
#include <core_cmFunc.h>         /* Core Function Access           */

#endif /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

/* check device defines and use defaults */

#if defined __CHECK_DEVICE_DEFINES

```

```

#ifndef __CM0PLUS_REV
#define __CM0PLUS_REV      0x0000
#warning "__CM0PLUS_REV not defined in device header file; using default!"
#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT      0
#warning "__MPU_PRESENT not defined in device header file; using default!"
#endif

#ifndef __VTOR_PRESENT
#define __VTOR_PRESENT      0
#warning "__VTOR_PRESENT not defined in device header file; using default!"
#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      2
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig  0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
\defgroup CMSIS_glob_defs CMSIS Global Defines

```

```

<strong>IO Type Qualifiers</strong> are used

\li to specify the access to peripheral variables.

\li for automatic generation of peripheral register debug information.

*/
#ifndef __cplusplus
#define __I volatile /*!< Defines 'read only' permissions */
#ifndef __cplusplus
#define __I volatile const /*!< Defines 'read only' permissions */
#endif
#define __O volatile /*!< Defines 'write only' permissions */
#define __IO volatile /*!< Defines 'read / write' permissions */

/*@} end of group Cortex-M0+ */

```

```

*****
*      Register Abstraction
Core Register contain:
- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core MPU Register
*****
/** \defgroup CMSIS_core_register Defines and Type Definitions
\brief Type definitions and defines for Cortex-M processor based devices.
*/

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.

@{

*/
/* brief Union type to access the Application Program Status Register (APSR).
 */
typedef union
{
    struct
    {
        #if (__CORTEX_M != 0x04)
            uint32_t _reserved0:27;      /*!< bit: 0..26 Reserved          */
        #else
            uint32_t _reserved0:16;      /*!< bit: 0..15 Reserved          */
            uint32_t GE:4;              /*!< bit: 16..19 Greater than or Equal flags */
            uint32_t _reserved1:7;      /*!< bit: 20..26 Reserved          */
        #endif
            uint32_t Q:1;                /*!< bit: 27 Saturation condition flag   */
            uint32_t V:1;                /*!< bit: 28 Overflow condition code flag */
            uint32_t C:1;                /*!< bit: 29 Carry condition code flag   */
            uint32_t Z:1;                /*!< bit: 30 Zero condition code flag    */
            uint32_t N:1;                /*!< bit: 31 Negative condition code flag */
        } b;                      /*!< Structure used for bit access       */
        uint32_t w;                /*!< Type     used for word access       */
    } APSR_Type;
}

```

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;               /*!< Type used for word access */
} IPSR_Type;

```

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        #if (__CORTEX_M != 0x04)
            uint32_t _reserved0:15; /*!< bit: 9..23 Reserved */
        #else
            uint32_t _reserved0:7; /*!< bit: 9..15 Reserved */
            uint32_t GE:4;         /*!< bit: 16..19 Greater than or Equal flags */
            uint32_t _reserved1:4; /*!< bit: 20..23 Reserved */
        #endif
        uint32_t T:1;             /*!< bit: 24 Thumb bit (read 0) */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;               /*!< Type used for word access */
} xPSR_Type;

```

```

    uint32_t IT:2;           /*!< bit: 25..26 saved IT state (read 0) */
    uint32_t Q:1;            /*!< bit: 27 Saturation condition flag */
    uint32_t V:1;            /*!< bit: 28 Overflow condition code flag */
    uint32_t C:1;            /*!< bit: 29 Carry condition code flag */
    uint32_t Z:1;            /*!< bit: 30 Zero condition code flag */
    uint32_t N:1;            /*!< bit: 31 Negative condition code flag */
} b;                      /*!< Structure used for bit access */
uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

```

```

/** \brief Union type to access the Control Registers (CONTROL).
*/
typedef union
{
    struct
    {
        uint32_t nPRIV:1;          /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} CONTROL_Type;

```

```
/*@} end of group CMSIS_CORE */
```

```
/** \ingroup CMSIS_core_register
```

```

\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers

@{
*

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[1];           /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */
    uint32_t RESERVED0[31];
    __IO uint32_t ICER[1];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
    uint32_t RSERVED1[31];
    __IO uint32_t ISPR[1];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */
    uint32_t RESERVED2[31];
    __IO uint32_t ICPR[1];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */
    uint32_t RESERVED3[31];
    uint32_t RESERVED4[64];
    __IO uint32_t IP[8];             /*!< Offset: 0x300 (R/W) Interrupt Priority Register */
} NVIC_Type;

/*@} end of group CMSIS_NVIC */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers

@{
*


```

```

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct
{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
} __attribute__((packed))

#if (__VTOR_PRESENT == 1)
    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register   */
#else
    uint32_t RESERVED0;
#endif

__attribute__((packed))
__IO uint32_t AIRCR;           /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register      */
__IO uint32_t SCR;             /*!< Offset: 0x010 (R/W) System Control Register        */
__IO uint32_t CCR;             /*!< Offset: 0x014 (R/W) Configuration Control Register */
uint32_t RESERVED1;

__IO uint32_t SHP[2];          /*!< Offset: 0x01C (R/W) System Handlers Priority Registers. [0] is
RESERVED      */
__IO uint32_t SHCSR;           /*!< Offset: 0x024 (R/W) System Handler Control and State Register
*/
} SCB_Type;

```

```
/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24                  /*!< SCB CPUID:  
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk     (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)  /*!<  
SCB CPUID: IMPLEMENTER Mask */
```

```

#define SCB_CPUID_VARIANT_Pos      20          /*!< SCB CPUID: VARIANT Position
*/
#define SCB_CPUID_VARIANT_Msk      (0xFUL << SCB_CPUID_VARIANT_Pos)    /*!< SCB CPUID:
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos  16          /*!< SCB CPUID:
ARCHITECTURE Position */
#define SCB_CPUID_ARCHITECTURE_Msk  (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)  /*!< SCB
CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos        4           /*!< SCB CPUID: PARTNO Position
*/
#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos)    /*!< SCB
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos      0           /*!< SCB CPUID: REVISION Position
*/
#define SCB_CPUID_REVISION_Msk      (0xFUL << SCB_CPUID_REVISION_Pos)    /*!< SCB CPUID:
REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos    31          /*!< SCB ICSR: NMIPENDSET
Position */
#define SCB_ICSR_NMIPENDSET_Msk    (1UL << SCB_ICSR_NMIPENDSET_Pos)    /*!< SCB ICSR:
NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos     28          /*!< SCB ICSR: PENDSVSET
Position */
#define SCB_ICSR_PENDSVSET_Msk     (1UL << SCB_ICSR_PENDSVSET_Pos)    /*!< SCB ICSR:
PENDSVSET Mask */

#define SCB_ICSR_PENDSVCLR_Pos    27          /*!< SCB ICSR: PENDSVCLR
Position */

```

#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */
#define SCB_ICSR_VECTPENDING_Pos	12	/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk	(0x1FFUL << SCB_ICSR_VECTPENDING_Pos)	/*!< SCB ICSR: VECTPENDING Mask */
#define SCB_ICSR_VECTACTIVE_Pos	0	/*!< SCB ICSR: VECTACTIVE Position */
#define SCB_ICSR_VECTACTIVE_Msk	(0x1FFUL << SCB_ICSR_VECTACTIVE_Pos)	/*!< SCB ICSR: VECTACTIVE Mask */

```

#if (__VTOR_PRESENT == 1)

/* SCB Interrupt Control State Register Definitions */

#define SCB_VTOR_TBLOFF_Pos      8          /*!< SCB VTOR: TBLOFF Position */
#define SCB_VTOR_TBLOFF_Msk      (0xFFFFFUL << SCB_VTOR_TBLOFF_Pos)    /*!< SCB VTOR:
TBLOFF Mask */

#endif

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos    16         /*!< SCB AIRCR: VECTKEY Position */
#define SCB_AIRCR_VECTKEY_Msk    (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)    /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos 16         /*!< SCB AIRCR: VECTKEYSTAT
Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)  /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos   15         /*!< SCB AIRCR: ENDIANESS
Position */
#define SCB_AIRCR_ENDIANESS_Msk   (1UL << SCB_AIRCR_ENDIANESS_Pos)     /*!< SCB
AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos 2          /*!< SCB AIRCR: SYSRESETREQ
Position */
#define SCB_AIRCR_SYSRESETREQ_Msk (1UL << SCB_AIRCR_SYSRESETREQ_Pos)   /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos 1        /*!< SCB AIRCR: VECTCLRACTIVE
Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos) /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

```

```

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos      4          /*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk      (1UL << SCB_SCR_SEVONPEND_Pos)      /*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos     2          /*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk     (1UL << SCB_SCR_SLEEPDEEP_Pos)      /*!< SCB SCR: SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos    1          /*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk    (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos      9          /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk      (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR: STKALIGN Mask */

#define SCB_CCR_UNALIGN_TRP_Pos    3          /*!< SCB CCR: UNALIGN_TRP Position */
#define SCB_CCR_UNALIGN_TRP_Msk    (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB CCR: UNALIGN_TRP Mask */

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_SVCALLPENDED_Pos 15         /*!< SCB SHCSR: SVCALLPENDED Position */
#define SCB_SHCSR_SVCALLPENDED_Msk (1UL << SCB_SHCSR_SVCALLPENDED_Pos)      /*!< SCB SHCSR: SVCALLPENDED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SysTick System Tick Timer (SysTick)
\brief Type definitions for the System Timer Registers.

@{

*/
/* \brief Structure type to access the System Timer (SysTick).
*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I  uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
} SysTick_Type;

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos     16           /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk     (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2            /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk     (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

```

```

#define SysTick_CTRL_TICKINT_Pos      1                  /*!< SysTick CTRL: TICKINT Position */
/*/

#define SysTick_CTRL_TICKINT_Msk     (1UL << SysTick_CTRL_TICKINT_Pos)    /*!< SysTick CTRL:
TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos      0                  /*!< SysTick CTRL: ENABLE Position */
/*/

#define SysTick_CTRL_ENABLE_Msk     (1UL << SysTick_CTRL_ENABLE_Pos)    /*!< SysTick
CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos      0                  /*!< SysTick LOAD: RELOAD
Position */

#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFUL << SysTick_LOAD_RELOAD_Pos) /*!<
SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos      0                  /*!< SysTick VAL: CURRENT
Position */

#define SysTick_VAL_CURRENT_Msk     (0xFFFFFUL << SysTick_VAL_CURRENT_Pos) /*!<
SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos      31                 /*!< SysTick CALIB: NOREF Position */
/*/

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)    /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos       30                 /*!< SysTick CALIB: SKEW Position */
/*/

#define SysTick_CALIB_SKEW_Msk     (1UL << SysTick_CALIB_SKEW_Pos)    /*!< SysTick CALIB:
SKEW Mask */

```

```

#define SysTick_CALIB_TENMS_Pos      0           /*!< SysTick CALIB: TENMS Position
*/
#define SysTick_CALIB_TENMS_Msk      (0xFFFFFUL << SysTick_VAL_CURRENT_Pos) /*!< SysTick
CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

#ifndef __MPU_PRESENT == 1
/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUs Memory Protection Unit (MPU)
\brief Type definitions for the Memory Protection Unit (MPU)

@{
*/
/* \brief Structure type to access the Memory Protection Unit (MPU).
*/
typedef struct
{
    __I uint32_t TYPE;          /*!< Offset: 0x000 (R/ ) MPU Type Register */
    __IO uint32_t CTRL;         /*!< Offset: 0x004 (R/W) MPU Control Register */
    __IO uint32_t RNR;          /*!< Offset: 0x008 (R/W) MPU Region RNRLber Register */
    __IO uint32_t RBAR;         /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
    __IO uint32_t RASR;         /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register
*/
} MPU_Type;

/* MPU Type Register */
#define MPU_TYPE_IREGION_Pos       16          /*!< MPU TYPE: IREGION Position
*/

```

```

#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)      /*!< MPU TYPE:  
IREGION Mask */

#define MPU_TYPE_DREGION_Pos     8                                /*!< MPU TYPE: DREGION Position */  
*/  
  
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)      /*!< MPU  
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0                                /*!< MPU TYPE: SEPARATE  
Position */  
  
#define MPU_TYPE_SEPARATE_Msk      (1UL << MPU_TYPE_SEPARATE_Pos)      /*!< MPU  
TYPE: SEPARATE Mask */

/* MPU Control Register */  
  
#define MPU_CTRL_PRIVDEFENA_Pos    2                                /*!< MPU CTRL: PRIVDEFENA  
Position */  
  
#define MPU_CTRL_PRIVDEFENA_Msk      (1UL << MPU_CTRL_PRIVDEFENA_Pos)      /*!< MPU  
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos     1                                /*!< MPU CTRL: HFNMIENA  
Position */  
  
#define MPU_CTRL_HFNMIENA_Msk      (1UL << MPU_CTRL_HFNMIENA_Pos)      /*!< MPU  
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos       0                                /*!< MPU CTRL: ENABLE Position */  
  
#define MPU_CTRL_ENABLE_Msk      (1UL << MPU_CTRL_ENABLE_Pos)      /*!< MPU CTRL:  
ENABLE Mask */

/* MPU Region Number Register */  
  
#define MPU_RNR_REGION_Pos       0                                /*!< MPU RNR: REGION Position */  
  
#define MPU_RNR_REGION_Msk      (0xFFUL << MPU_RNR_REGION_Pos)      /*!< MPU RNR:  
REGION Mask */

```

```

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos      8          /*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk      (0xFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos     4          /*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk     (1UL << MPU_RBAR_VALID_Pos)        /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos    0          /*!< MPU RBAR: REGION Position */
*/
#define MPU_RBAR_REGION_Msk    (0xFUL << MPU_RBAR_REGION_Pos)      /*!< MPU
RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos     16         /*!< MPU RASR: MPU Region
Attribute field Position */
#define MPU_RASR_ATTRS_Msk     (0xFFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos        28         /*!< MPU RASR: ATTRS.XN Position */
*/
#define MPU_RASR_XN_Msk        (1UL << MPU_RASR_XN_Pos)          /*!< MPU RASR:
ATTRS.XN Mask */

#define MPU_RASR_AP_Pos        24         /*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk        (0x7UL << MPU_RASR_AP_Pos)        /*!< MPU RASR:
ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos       19         /*!< MPU RASR: ATTRS.TEX Position */
*/
#define MPU_RASR_TEX_Msk       (0x7UL << MPU_RASR_TEX_Pos)        /*!< MPU RASR:
ATTRS.TEX Mask */

```

```

#define MPU_RASR_S_Pos           18          /*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk          (1UL << MPU_RASR_S_Pos)      /*!< MPU RASR: ATTRS.S
Mask */

#define MPU_RASR_C_Pos           17          /*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk          (1UL << MPU_RASR_C_Pos)      /*!< MPU RASR:
ATTRS.C Mask */

#define MPU_RASR_B_Pos           16          /*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk          (1UL << MPU_RASR_B_Pos)      /*!< MPU RASR:
ATTRS.B Mask */

#define MPU_RASR_SRD_Pos          8          /*!< MPU RASR: Sub-Region Disable
Position */
#define MPU_RASR_SRD_Msk         (0xFFUL << MPU_RASR_SRD_Pos)  /*!< MPU RASR:
Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos          1          /*!< MPU RASR: Region Size Field
Position */
#define MPU_RASR_SIZE_Msk        (0x1FUL << MPU_RASR_SIZE_Pos)  /*!< MPU RASR:
Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos          0          /*!< MPU RASR: Region enable bit
Position */
#define MPU_RASR_ENABLE_Msk       (1UL << MPU_RASR_ENABLE_Pos)  /*!< MPU RASR:
Region enable bit Disable Mask */

/*}@} end of group CMSIS_MPU */
#endif

```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug Core Debug Registers (CoreDebug)
\brief Cortex-M0+ Core Debug Registers (DCB registers, SHCSR, and DFSR)
are only accessible over DAP and not via processor. Therefore
they are not covered by the Cortex-M0 header file.
```

```
@{
*/
/*@} end of group CMSIS_CoreDebug */
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_core_base Core Definitions
\brief Definitions for base addresses, unions, and structures.
@{
*/

```

```
/* Memory mapping of Cortex-M0+ Hardware */

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */
#define SysTick_BASE  (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */
#define NVIC_BASE    (SCS_BASE + 0x0100UL)        /*!< NVIC Base Address */
#define SCB_BASE     (SCS_BASE + 0x0D00UL)        /*!< System Control Block Base Address */

#define SCB          ((SCB_Type *) SCB_BASE)      /*!< SCB configuration struct */
#define SysTick      ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */
#define NVIC         ((NVIC_Type *) NVIC_BASE)     /*!< NVIC configuration struct */

#if (__MPU_PRESENT == 1)
#define MPU_BASE     (SCS_BASE + 0x0D90UL)        /*!< Memory Protection Unit */
#define MPU          ((MPU_Type *) MPU_BASE)       /*!< Memory Protection Unit */

```

```

#endif

/*}@*/



/*********************  

*      Hardware Abstraction Layer  

Core Function Interface contains:  

- Core NVIC Functions  

- Core SysTick Functions  

- Core Register Access Functions  

*****/  

/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference  

*/



/* ##### NVIC functions ##### */  

/** \ingroup CMSIS_Core_FunctionInterface  

\defgroup CMSIS_Core_NVICFunctions NVIC Functions  

\brief Functions that manage interrupts and exceptions via the NVIC.  

@{  

*/



/* Interrupt Priorities are WORD accessible only under ARMv6M */  

/* The following MACROS handle generation of the register offset and byte masks */  

#define _BIT_SHIFT(IRQn)    ( (((uint32_t)(IRQn)) & 0x03) * 8 )  

#define _SHP_IDX(IRQn)     ( (((uint32_t)(IRQn) & 0x0F)-8) >> 2 )
```

```
#define _IP_IDX(IRQn)      ((uint32_t)(IRQn)      >> 2)
```

```
/** \brief Enable External Interrupt
```

The function enables a device-specific interrupt in the NVIC interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
_STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ISER[0] = (1 << ((uint32_t)(IRQn) & 0x1F));
```

```
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
_STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ICER[0] = (1 << ((uint32_t)(IRQn) & 0x1F));
```

```
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

*/

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
{
    return((uint32_t) ((NVIC->ISPR[0] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0));
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

\param [in] IRQn Interrupt number. Value cannot be negative.

*/

```
__STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ISPR[0] = (1 << ((uint32_t)(IRQn) & 0x1F));
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```

\param [in] IRQn External interrupt number. Value cannot be negative.

*/
__STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ICPR[0] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* Clear pending interrupt */
}

```

/** \brief Set Interrupt Priority

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

```

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

*/
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
    if(IRQn < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = (SCB->SHP[_SHP_IDX(IRQn)] & ~(_BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn));
    } else {
        NVIC->IP[_IP_IDX(IRQn)] = (NVIC->IP[_IP_IDX(IRQn)] & ~(_BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn));
    }
}
```

```
/** \brief Get Interrupt Priority
```

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

\param [in] IRQn Interrupt number.

\return Interrupt Priority. Value is aligned automatically to the implemented priority bits of the microcontroller.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{
    if(IRQn < 0) {
        return((uint32_t)((SCB->SHP[_SHP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & 0xFF) >> (8 - __NVIC_PRIO_BITS)); /* get priority for Cortex-M0 system interrupts */
    }
    else {
        return((uint32_t)((NVIC->IP[_IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & 0xFF) >> (8 - __NVIC_PRIO_BITS)); /* get priority for device specific interrupts */
    }
}
```

```
/** \brief System Reset
```

The function initiates a system reset request to reset the MCU.

```
*/
```

```
__STATIC_INLINE void NVIC_SystemReset(void)
{
```

```

__DSB();                                /* Ensure all outstanding memory accesses included
                                         buffered write are completed before reset */

SCB->AIRCR = ((0x5FA << SCB_AIRCR_VECTKEY_Pos) |

    SCB_AIRCR_SYSRESETREQ_Msk);

__DSB();                                /* Ensure completion of memory access */

while(1);                                /* wait until reset */

}

/*@} end of CMSIS_Core_NVICFunctions */

```

/ ##### SysTick function
/

```

/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.

 @{
 */

```

```
#if (__Vendor_SysTickConfig == 0)
```

```
/** \brief System Tick Configuration
```

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.

Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

```
\return 0 Function succeeded.
```

```
\return 1 Function failed.
```

\note When the variable __Vendor_SysTickConfig is set to 1, then the function SysTick_Config is not included. In this case, the file <i>device</i>.h must contain a vendor-specific implementation of this function.

```
*/
```

```
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1) > SysTick_LOAD_RELOAD_Msk) return (1); /* Reload value impossible */

    SysTick->LOAD = ticks - 1; /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); /* set Priority for Systick Interrupt */
    SysTick->VAL = 0; /* Load the SysTick Counter Value */
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
        SysTick_CTRL_TICKINT_Msk |
        SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */
    return (0); /* Function successful */
}
```

```
#endif
```

```
/*@} end of CMSIS_Core_SysTickFunctions */
```

```
#endif /* __CORE_CM0PLUS_H_DEPENDANT */
```

```
#ifdef __cplusplus
}

#endif /* __CMSIS_GENERIC */

/*
** #####
** Processors:      MKL25Z128FM4
**
**                  MKL25Z128FT4
**
**                  MKL25Z128LH4
**
**                  MKL25Z128VLK4
**
**
** Compilers:      Keil ARM C/C++ Compiler
**
**                  Freescale C/C++ for Embedded ARM
**
**                  GNU C Compiler
**
**                  GNU C Compiler - CodeSourcery Sourcery G++
**
**                  IAR ANSI C/C++ Compiler for ARM
**
**
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012
**
** Version:        rev. 2.5, 2015-02-19
**
** Build:          b150220
**
**
** Abstract:
**
** CMSIS Peripheral Access Layer for MKL25Z4
**
**
** Copyright (c) 1997 - 2015 Freescale Semiconductor, Inc.
**
** All rights reserved.
**
**
```

** Redistribution and use in source and binary forms, with or without modification,
** are permitted provided that the following conditions are met:
**
** o Redistributions of source code must retain the above copyright notice, this list
** of conditions and the following disclaimer.
**
** o 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.
**
** o Neither the name of Freescale Semiconductor, Inc. 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.
**
** http: www.freescale.com
** mail: support@freescale.com
**
** Revisions:

** - rev. 1.0 (2012-06-13)

** Initial version.

** - rev. 1.1 (2012-06-21)

** Update according to reference manual rev. 1.

** - rev. 1.2 (2012-08-01)

** Device type UARTLP changed to UART0.

** - rev. 1.3 (2012-10-04)

** Update according to reference manual rev. 3.

** - rev. 1.4 (2012-11-22)

** MCG module - bit LOLS in MCG_S register renamed to LOLS0.

** NV registers - bit EZPORT_DIS in NV_FOPT register removed.

** - rev. 1.5 (2013-04-05)

** Changed start of doxygen comment.

** - rev. 2.0 (2013-10-29)

** Register accessor macros added to the memory map.

** Symbols for Processor Expert memory map compatibility added to the memory map.

** Startup file for gcc has been updated according to CMSIS 3.2.

** System initialization updated.

** - rev. 2.1 (2014-07-16)

** Module access macro module_BASES replaced by module_BASE_PTRS.

** System initialization and startup updated.

** - rev. 2.2 (2014-08-22)

** System initialization updated - default clock config changed.

** - rev. 2.3 (2014-08-28)

** Update of startup files - possibility to override DefaultISR added.

** - rev. 2.4 (2014-10-14)

** Interrupt INT_LPTimer renamed to INT_LPTMR0.

** - rev. 2.5 (2015-02-19)

** Renamed interrupt vector LLW to LLWU.

```

**  

** #####  

*/  

  

/*!  

 * @file MKL25Z4.h  

 * @version 2.5  

 * @date 2015-02-19  

 * @brief CMSIS Peripheral Access Layer for MKL25Z4  

 *  

 * CMSIS Peripheral Access Layer for MKL25Z4  

*/  

  

/* -----  

-- MCU activation  

----- */  

  

/* Prevention from multiple including the same memory map */  

#if !defined(MKL25Z4_H_) /* Check if memory map has not been already included */  

#define MKL25Z4_H_  

#define MCU_MKL25Z4  

  

/* Check if another memory map has not been also included */  

#if (defined(MCU_ACTIVE))  

    #error MKL25Z4 memory map: There is already included another memory map. Only one memory map  

can be included.  

#endif /* (defined(MCU_ACTIVE)) */  

#define MCU_ACTIVE

```

```

#include <stdint.h>

/** Memory map major version (memory maps with equal major version number are
 * compatible) */

#define MCU_MEM_MAP_VERSION 0x0200u

/** Memory map minor version */

#define MCU_MEM_MAP_VERSION_MINOR 0x0005u

/*
----- --
-- Interrupt vector numbers
----- */
/*!

* @addtogroup Interrupt_vector_numbers Interrupt vector numbers
* @{
*/
/*!

** Interrupt Number Definitions */

#define NUMBER_OF_INT_VECTORS 48           /**< Number of interrupts in the Vector table */

typedef enum IRQn {
    /* Core interrupts */
    NonMaskableInt_IRQn      = -14,      /**< Non Maskable Interrupt */
    HardFault_IRQn            = -13,      /**< Cortex-M0 SV Hard Fault Interrupt */
    SVCall_IRQn               = -5,       /**< Cortex-M0 SV Call Interrupt */
    PendSV_IRQn               = -2,       /**< Cortex-M0 Pend SV Interrupt */
    SysTick_IRQn               = -1,      /**< Cortex-M0 System Tick Interrupt */

```

```

/* Device specific interrupts */

DMA0_IRQn      = 0,          /**< DMA channel 0 transfer complete */
DMA1_IRQn      = 1,          /**< DMA channel 1 transfer complete */
DMA2_IRQn      = 2,          /**< DMA channel 2 transfer complete */
DMA3_IRQn      = 3,          /**< DMA channel 3 transfer complete */
Reserved20_IRQn = 4,          /**< Reserved interrupt */
FTFA_IRQn      = 5,          /**< Command complete and read collision */
LVD_LVW_IRQn   = 6,          /**< Low-voltage detect, low-voltage warning */
LLWU_IRQn      = 7,          /**< Low leakage wakeup Unit */
I2C0_IRQn      = 8,          /**< I2C0 interrupt */
I2C1_IRQn      = 9,          /**< I2C1 interrupt */
SPI0_IRQn      = 10,         /**< SPI0 single interrupt vector for all sources */
SPI1_IRQn      = 11,         /**< SPI1 single interrupt vector for all sources */
UART0_IRQn     = 12,         /**< UART0 status and error */
UART1_IRQn     = 13,         /**< UART1 status and error */
UART2_IRQn     = 14,         /**< UART2 status and error */
ADC0_IRQn      = 15,         /**< ADC0 interrupt */
CMPO_IRQn      = 16,         /**< CMPO interrupt */
TPM0_IRQn      = 17,         /**< TPM0 single interrupt vector for all sources */
TPM1_IRQn      = 18,         /**< TPM1 single interrupt vector for all sources */
TPM2_IRQn      = 19,         /**< TPM2 single interrupt vector for all sources */
RTC_IRQn       = 20,         /**< RTC alarm */
RTC_Seconds_IRQn = 21,        /**< RTC seconds */
PIT_IRQn       = 22,         /**< PIT interrupt */
Reserved39_IRQn = 23,        /**< Reserved interrupt */
USB0_IRQn      = 24,         /**< USB0 interrupt */
DAC0_IRQn      = 25,         /**< DAC0 interrupt */
TSI0_IRQn      = 26,         /**< TSI0 interrupt */

```

```

MCG_IRQn          = 27,      /**< MCG interrupt */
LPTMR0_IRQn       = 28,      /**< LPTMR0 interrupt */
Reserved45_IRQn   = 29,      /**< Reserved interrupt */
PORTA_IRQn        = 30,      /**< PORTA Pin detect */
PORTD_IRQn        = 31,      /**< PORTD Pin detect */

} IRQn_Type;

```

```

/*!
* @}
* // /* end of group Interrupt_vector_numbers */

```

```

/*
-- Cortex M0 Core Configuration
----- */

```

```

/*!
* @addtogroup Cortex_Core_Configuration Cortex M0 Core Configuration
* @{
*/

```

```

#define __CM0PLUS_REV          0x0000  /**< Core revision r0p0 */
#define __MPU_PRESENT           0      /**< Defines if an MPU is present or not */
#define __VTOR_PRESENT          1      /**< Defines if an MPU is present or not */
#define __NVIC_PRIO_BITS         2      /**< Number of priority bits implemented in the NVIC */
#define __Vendor_SysTickConfig    0      /**< Vendor specific implementation of SysTickConfig is
defined */

```

```

#include "core_cm0plus.h"      /* Core Peripheral Access Layer */

```

```
#include "system_MKL25Z4.h"      /* Device specific configuration file */

/*!
* @}
* /* end of group Cortex_Core_Configuration */

/*
-----  

-- Device Peripheral Access Layer  

----- */
/*!  

* @addtogroup Peripheral_access_layer Device Peripheral Access Layer
* @{
* /  

  

/*  

** Start of section using anonymous unions  

*/
  

#if defined(__ARMCC_VERSION)
#pragma push
#pragma anon_unions
#elif defined(__CWCC__)
#pragma push
#pragma cpp_extensions on
#elif defined(__GNUC__)
/* anonymous unions are enabled by default */

```

```

#ifndef __IAR_SYSTEMS_ICC__
#define __IAR_SYSTEMS_ICC__ 1
#endif

#ifndef __IAR_SYSTEMS_ICC__
#pragma language=extended
#endif

#ifndef __IAR_SYSTEMS_ICC__
#error Not supported compiler type
#endif

/* -----
-- ADC Peripheral Access Layer
----- */

/*!
* @addtogroup ADC_Peripheral_Access_Layer ADC Peripheral Access Layer
* @{
*/

```

/*! **ADC - Register Layout Typedef */**

```

typedef struct {
    __IO uint32_t SC1[2];           /**<< ADC Status and Control Registers 1, array offset: 0x0, array step: 0x4 */
    __IO uint32_t CFG1;             /**<< ADC Configuration Register 1, offset: 0x8 */
    __IO uint32_t CFG2;             /**<< ADC Configuration Register 2, offset: 0xC */
    __I uint32_t R[2];              /**<< ADC Data Result Register, array offset: 0x10, array step: 0x4 */
    __IO uint32_t CV1;              /**<< Compare Value Registers, offset: 0x18 */
    __IO uint32_t CV2;              /**<< Compare Value Registers, offset: 0x1C */
    __IO uint32_t SC2;              /**<< Status and Control Register 2, offset: 0x20 */
    __IO uint32_t SC3;              /**<< Status and Control Register 3, offset: 0x24 */
    __IO uint32_t OFS;              /**<< ADC Offset Correction Register, offset: 0x28 */
    __IO uint32_t PG;               /**<< ADC Plus-Side Gain Register, offset: 0x2C */
    __IO uint32_t MG;               /**<< ADC Minus-Side Gain Register, offset: 0x30 */
}

```

```

__IO uint32_t CLPD;           /**< ADC Plus-Side General Calibration Value Register, offset:  
0x34 */

__IO uint32_t CLPS;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x38  
*/  
*/  
__IO uint32_t CLP4;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x3C  
*/  
__IO uint32_t CLP3;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x40  
*/  
__IO uint32_t CLP2;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x44  
*/  
__IO uint32_t CLP1;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x48  
*/  
__IO uint32_t CLP0;           /**< ADC Plus-Side General Calibration Value Register, offset: 0x4C  
*/  
  
    uint8_t RESERVED_0[4];  
  
__IO uint32_t CLMD;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x54 */  
  
__IO uint32_t CLMS;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x58 */  
  
__IO uint32_t CLM4;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x5C */  
  
__IO uint32_t CLM3;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x60 */  
  
__IO uint32_t CLM2;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x64 */  
  
__IO uint32_t CLM1;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x68 */  
  
__IO uint32_t CLM0;           /**< ADC Minus-Side General Calibration Value Register, offset:  
0x6C */  
  
} ADC_Type, *ADC_MemMapPtr;  
  
/* -----  
-- ADC - Register accessor macros  
----- */
```

```

/*!

* @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros

* @{
* /


/* ADC - Register accessors */

#define ADC_SC1_REG(base,index)      (((base)->SC1[index]))
#define ADC_SC1_COUNT                2
#define ADC_CFG1_REG(base)           (((base)->CFG1))
#define ADC_CFG2_REG(base)           (((base)->CFG2))
#define ADC_R_REG(base,index)        (((base)->R[index]))
#define ADC_R_COUNT                  2
#define ADC_CV1_REG(base)            (((base)->CV1))
#define ADC_CV2_REG(base)            (((base)->CV2))
#define ADC_SC2_REG(base)            (((base)->SC2))
#define ADC_SC3_REG(base)            (((base)->SC3))
#define ADC_OFS_REG(base)            (((base)->OFS))
#define ADC_PG_REG(base)             (((base)->PG))
#define ADC_MG_REG(base)             (((base)->MG))
#define ADC_CLPD_REG(base)           (((base)->CLPD))
#define ADC_CLPS_REG(base)           (((base)->CLPS))
#define ADC_CLP4_REG(base)           (((base)->CLP4))
#define ADC_CLP3_REG(base)           (((base)->CLP3))
#define ADC_CLP2_REG(base)           (((base)->CLP2))
#define ADC_CLP1_REG(base)           (((base)->CLP1))
#define ADC_CLP0_REG(base)           (((base)->CLP0))
#define ADC_CLMD_REG(base)           (((base)->CLMD))

```

```
#define ADC_CLMS_REG(base) ((base)->CLMS)
#define ADC_CLM4_REG(base) ((base)->CLM4)
#define ADC_CLM3_REG(base) ((base)->CLM3)
#define ADC_CLM2_REG(base) ((base)->CLM2)
#define ADC_CLM1_REG(base) ((base)->CLM1)
#define ADC_CLM0_REG(base) ((base)->CLM0)
```

```
/*!
 * @}
 */
/* /* end of group ADC_Register_Accessor_Macros */
```

```
/*
-----  
-- ADC Register Masks  
----- */
```

```
/*!
 * @addtogroup ADC_Register_Masks ADC Register Masks
 * @{
 */

```

```
/* SC1 Bit Fields */
#define ADC_SC1_ADCH_MASK 0x1Fu
#define ADC_SC1_ADCH_SHIFT 0
#define ADC_SC1_ADCH_WIDTH 5
#define ADC_SC1_ADCH(x) (((uint32_t)((uint32_t)(x)<<ADC_SC1_ADCH_SHIFT))&ADC_SC1_ADCH_MASK)
#define ADC_SC1_DIFF_MASK 0x20u
#define ADC_SC1_DIFF_SHIFT 5
```

```

#define ADC_SC1_DIFF_WIDTH          1
#define ADC_SC1_DIFF(x)             (((uint32_t)((uint32_t)(x)<<ADC_SC1_DIFF_SHIFT))&ADC_SC1_DIFF_MASK)
#define ADC_SC1_AIEN_MASK           0x40u
#define ADC_SC1_AIEN_SHIFT          6
#define ADC_SC1_AIEN_WIDTH          1
#define ADC_SC1_AIEN(x)             (((uint32_t)((uint32_t)(x)<<ADC_SC1_AIEN_SHIFT))&ADC_SC1_AIEN_MASK)
#define ADC_SC1_COCO_MASK            0x80u
#define ADC_SC1_COCO_SHIFT           7
#define ADC_SC1_COCO_WIDTH          1
#define ADC_SC1_COCO(x)              (((uint32_t)((uint32_t)(x)<<ADC_SC1_COCO_SHIFT))&ADC_SC1_COCO_MASK)

/* CFG1 Bit Fields */

#define ADC_CFG1_ADICLK_MASK         0x3u
#define ADC_CFG1_ADICLK_SHIFT          0
#define ADC_CFG1_ADICLK_WIDTH         2
#define ADC_CFG1_ADICLK(x)             (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADICLK_SHIFT))&ADC_CFG1_ADICLK_MASK)
#define ADC_CFG1_MODE_MASK             0xCu
#define ADC_CFG1_MODE_SHIFT            2
#define ADC_CFG1_MODE_WIDTH           2
#define ADC_CFG1_MODE(x)               (((uint32_t)((uint32_t)(x)<<ADC_CFG1_MODE_SHIFT))&ADC_CFG1_MODE_MASK)
#define ADC_CFG1_ADLSMP_MASK           0x10u
#define ADC_CFG1_ADLSMP_SHIFT          4
#define ADC_CFG1_ADLSMP_WIDTH          1
#define ADC_CFG1_ADLSMP(x)              (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADLSMP_SHIFT))&ADC_CFG1_ADLSMP_MASK)
#define ADC_CFG1_ADIV_MASK             0x60u
#define ADC_CFG1_ADIV_SHIFT             5

```

```

#define ADC_CFG1_ADIV_WIDTH          2
#define ADC_CFG1_ADIV(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADIV_SHIFT))&ADC_CFG1_ADIV_MASK)
#define ADC_CFG1_ADLPC_MASK          0x80u
#define ADC_CFG1_ADLPC_SHIFT         7
#define ADC_CFG1_ADLPC_WIDTH         1
#define ADC_CFG1_ADLPC(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADLPC_SHIFT))&ADC_CFG1_ADLPC_MASK)

/* CFG2 Bit Fields */

#define ADC_CFG2_ADLSTS_MASK          0x3u
#define ADC_CFG2_ADLSTS_SHIFT         0
#define ADC_CFG2_ADLSTS_WIDTH         2
#define ADC_CFG2_ADLSTS(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADLSTS_SHIFT))&ADC_CFG2_ADLSTS_MASK)

#define ADC_CFG2_ADHSC_MASK           0x4u
#define ADC_CFG2_ADHSC_SHIFT          2
#define ADC_CFG2_ADHSC_WIDTH          1
#define ADC_CFG2_ADHSC(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADHSC_SHIFT))&ADC_CFG2_ADHSC_MASK)

#define ADC_CFG2_ADACKEN_MASK          0x8u
#define ADC_CFG2_ADACKEN_SHIFT         3
#define ADC_CFG2_ADACKEN_WIDTH         1
#define ADC_CFG2_ADACKEN(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADACKEN_SHIFT))&ADC_CFG2_ADACKEN_MASK)

#define ADC_CFG2_MUXSEL_MASK           0x10u
#define ADC_CFG2_MUXSEL_SHIFT          4
#define ADC_CFG2_MUXSEL_WIDTH          1
#define ADC_CFG2_MUXSEL(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG2_MUXSEL_SHIFT))&ADC_CFG2_MUXSEL_MASK)

/* R Bit Fields */

#define ADC_R_D_MASK                 0xFFFFFu

```

```

#define ADC_R_D_SHIFT          0
#define ADC_R_D_WIDTH          16
#define ADC_R_D(x)             (((uint32_t)((uint32_t)(x))<<ADC_R_D_SHIFT))&ADC_R_D_MASK

/* CV1 Bit Fields */

#define ADC_CV1_CV_MASK        0xFFFFu
#define ADC_CV1_CV_SHIFT        0
#define ADC_CV1_CV_WIDTH        16
#define ADC_CV1_CV(x)           (((uint32_t)((uint32_t)(x))<<ADC_CV1_CV_SHIFT))&ADC_CV1_CV_MASK

/* CV2 Bit Fields */

#define ADC_CV2_CV_MASK        0xFFFFu
#define ADC_CV2_CV_SHIFT        0
#define ADC_CV2_CV_WIDTH        16
#define ADC_CV2_CV(x)           (((uint32_t)((uint32_t)(x))<<ADC_CV2_CV_SHIFT))&ADC_CV2_CV_MASK

/* SC2 Bit Fields */

#define ADC_SC2_REFSEL_MASK     0x3u
#define ADC_SC2_REFSEL_SHIFT    0
#define ADC_SC2_REFSEL_WIDTH    2
#define ADC_SC2_REFSEL(x)        (((uint32_t)((uint32_t)(x))<<ADC_SC2_REFSEL_SHIFT))&ADC_SC2_REFSEL_MASK

#define ADC_SC2_DMAEN_MASK      0x4u
#define ADC_SC2_DMAEN_SHIFT     2
#define ADC_SC2_DMAEN_WIDTH     1
#define ADC_SC2_DMAEN(x)         (((uint32_t)((uint32_t)(x))<<ADC_SC2_DMAEN_SHIFT))&ADC_SC2_DMAEN_MASK

#define ADC_SC2_ACREN_MASK      0x8u
#define ADC_SC2_ACREN_SHIFT     3
#define ADC_SC2_ACREN_WIDTH     1

```

```

#define ADC_SC2_ACREN(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACREN_SHIFT))&ADC_SC2_ACREN_MASK)

#define ADC_SC2_ACFGT_MASK          0x10u
#define ADC_SC2_ACFGT_SHIFT         4
#define ADC_SC2_ACFGT_WIDTH         1

#define ADC_SC2_ACFGT(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFGT_SHIFT))&ADC_SC2_ACFGT_MASK)

#define ADC_SC2_ACFE_MASK           0x20u
#define ADC_SC2_ACFE_SHIFT          5
#define ADC_SC2_ACFE_WIDTH          1

#define ADC_SC2_ACFE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFE_SHIFT))&ADC_SC2_ACFE_MASK)

#define ADC_SC2_ADTRG_MASK          0x40u
#define ADC_SC2_ADTRG_SHIFT         6
#define ADC_SC2_ADTRG_WIDTH         1

#define ADC_SC2_ADTRG(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADTRG_SHIFT))&ADC_SC2_ADTRG_MASK)

#define ADC_SC2_ADACT_MASK           0x80u
#define ADC_SC2_ADACT_SHIFT          7
#define ADC_SC2_ADACT_WIDTH          1

#define ADC_SC2_ADACT(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADACT_SHIFT))&ADC_SC2_ADACT_MASK)

/* SC3 Bit Fields */

#define ADC_SC3_AVGS_MASK           0x3u
#define ADC_SC3_AVGS_SHIFT           0
#define ADC_SC3_AVGS_WIDTH           2

#define ADC_SC3_AVGS(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGS_SHIFT))&ADC_SC3_AVGS_MASK)

#define ADC_SC3_AVGE_MASK            0x4u
#define ADC_SC3_AVGE_SHIFT           2
#define ADC_SC3_AVGE_WIDTH           1

```

```

#define ADC_SC3_AVGE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGE_SHIFT))&ADC_SC3_AVGE_MASK)

#define ADC_SC3_ADCO_MASK          0x8u
#define ADC_SC3_ADCO_SHIFT         3
#define ADC_SC3_ADCO_WIDTH         1

#define ADC_SC3_ADCO(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_ADCO_SHIFT))&ADC_SC3_ADCO_MASK)

#define ADC_SC3_CALF_MASK          0x40u
#define ADC_SC3_CALF_SHIFT         6
#define ADC_SC3_CALF_WIDTH         1

#define ADC_SC3_CALF(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CALF_SHIFT))&ADC_SC3_CALF_MASK)

#define ADC_SC3_CAL_MASK           0x80u
#define ADC_SC3_CAL_SHIFT          7
#define ADC_SC3_CAL_WIDTH          1

#define ADC_SC3_CAL(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CAL_SHIFT))&ADC_SC3_CAL_MASK)

/* OFS Bit Fields */

#define ADC_OFS_OFS_MASK           0xFFFFu
#define ADC_OFS_OFS_SHIFT          0
#define ADC_OFS_OFS_WIDTH          16

#define ADC_OFS_OFS(x)
(((uint32_t)((uint32_t)(x)<<ADC_OFS_OFS_SHIFT))&ADC_OFS_OFS_MASK)

/* PG Bit Fields */

#define ADC_PG_PG_MASK             0xFFFFu
#define ADC_PG_PG_SHIFT            0
#define ADC_PG_PG_WIDTH            16

#define ADC_PG_PG(x)
(((uint32_t)((uint32_t)(x)<<ADC_PG_PG_SHIFT))&ADC_PG_PG_MASK)

/* MG Bit Fields */

#define ADC_MG_MG_MASK             0xFFFFu

```

```

#define ADC_MG_MG_SHIFT          0
#define ADC_MG_MG_WIDTH          16
#define ADC_MG_MG(x) (((uint32_t)((uint32_t)(x))<<ADC_MG_MG_SHIFT))&ADC_MG_MG_MASK)

/* CLPD Bit Fields */

#define ADC_CLPD_CLPD_MASK        0x3Fu
#define ADC_CLPD_CLPD_SHIFT       0
#define ADC_CLPD_CLPD_WIDTH       6
#define ADC_CLPD_CLPD(x) (((uint32_t)((uint32_t)(x))<<ADC_CLPD_CLPD_SHIFT))&ADC_CLPD_CLPD_MASK)

/* CLPS Bit Fields */

#define ADC_CLPS_CLPS_MASK        0x3Fu
#define ADC_CLPS_CLPS_SHIFT       0
#define ADC_CLPS_CLPS_WIDTH       6
#define ADC_CLPS_CLPS(x) (((uint32_t)((uint32_t)(x))<<ADC_CLPS_CLPS_SHIFT))&ADC_CLPS_CLPS_MASK)

/* CLP4 Bit Fields */

#define ADC_CLP4_CLP4_MASK        0x3FFu
#define ADC_CLP4_CLP4_SHIFT       0
#define ADC_CLP4_CLP4_WIDTH       10
#define ADC_CLP4_CLP4(x) (((uint32_t)((uint32_t)(x))<<ADC_CLP4_CLP4_SHIFT))&ADC_CLP4_CLP4_MASK)

/* CLP3 Bit Fields */

#define ADC_CLP3_CLP3_MASK        0x1FFu
#define ADC_CLP3_CLP3_SHIFT       0
#define ADC_CLP3_CLP3_WIDTH       9
#define ADC_CLP3_CLP3(x) (((uint32_t)((uint32_t)(x))<<ADC_CLP3_CLP3_SHIFT))&ADC_CLP3_CLP3_MASK)

/* CLP2 Bit Fields */

#define ADC_CLP2_CLP2_MASK        0xFFu

```

```

#define ADC_CLP2_CLP2_SHIFT          0
#define ADC_CLP2_CLP2_WIDTH          8
#define ADC_CLP2_CLP2(x)             (((uint32_t)((uint32_t)(x)<<ADC_CLP2_CLP2_SHIFT))&ADC_CLP2_CLP2_MASK)

/* CLP1 Bit Fields */

#define ADC_CLP1_CLP1_MASK           0x7Fu
#define ADC_CLP1_CLP1_SHIFT          0
#define ADC_CLP1_CLP1_WIDTH          7
#define ADC_CLP1_CLP1(x)             (((uint32_t)((uint32_t)(x)<<ADC_CLP1_CLP1_SHIFT))&ADC_CLP1_CLP1_MASK)

/* CLP0 Bit Fields */

#define ADC_CLP0_CLP0_MASK           0x3Fu
#define ADC_CLP0_CLP0_SHIFT          0
#define ADC_CLP0_CLP0_WIDTH          6
#define ADC_CLP0_CLP0(x)             (((uint32_t)((uint32_t)(x)<<ADC_CLP0_CLP0_SHIFT))&ADC_CLP0_CLP0_MASK)

/* CLMD Bit Fields */

#define ADC_CLMD_CLMD_MASK           0x3Fu
#define ADC_CLMD_CLMD_SHIFT          0
#define ADC_CLMD_CLMD_WIDTH          6
#define ADC_CLMD_CLMD(x)             (((uint32_t)((uint32_t)(x)<<ADC_CLMD_CLMD_SHIFT))&ADC_CLMD_CLMD_MASK)

/* CLMS Bit Fields */

#define ADC_CLMS_CLMS_MASK           0x3Fu
#define ADC_CLMS_CLMS_SHIFT          0
#define ADC_CLMS_CLMS_WIDTH          6
#define ADC_CLMS_CLMS(x)             (((uint32_t)((uint32_t)(x)<<ADC_CLMS_CLMS_SHIFT))&ADC_CLMS_CLMS_MASK)

/* CLM4 Bit Fields */

#define ADC_CLM4_CLM4_MASK           0x3FFu

```

```

#define ADC_CLM4_CLM4_SHIFT          0
#define ADC_CLM4_CLM4_WIDTH          10
#define ADC_CLM4_CLM4(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM4_CLM4_SHIFT))&ADC_CLM4_CLM4_MASK)

/* CLM3 Bit Fields */

#define ADC_CLM3_CLM3_MASK           0x1FFu
#define ADC_CLM3_CLM3_SHIFT          0
#define ADC_CLM3_CLM3_WIDTH          9
#define ADC_CLM3_CLM3(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM3_CLM3_SHIFT))&ADC_CLM3_CLM3_MASK)

/* CLM2 Bit Fields */

#define ADC_CLM2_CLM2_MASK           0xFFu
#define ADC_CLM2_CLM2_SHIFT          0
#define ADC_CLM2_CLM2_WIDTH          8
#define ADC_CLM2_CLM2(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM2_CLM2_SHIFT))&ADC_CLM2_CLM2_MASK)

/* CLM1 Bit Fields */

#define ADC_CLM1_CLM1_MASK           0x7Fu
#define ADC_CLM1_CLM1_SHIFT          0
#define ADC_CLM1_CLM1_WIDTH          7
#define ADC_CLM1_CLM1(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM1_CLM1_SHIFT))&ADC_CLM1_CLM1_MASK)

/* CLM0 Bit Fields */

#define ADC_CLM0_CLM0_MASK           0x3Fu
#define ADC_CLM0_CLM0_SHIFT          0
#define ADC_CLM0_CLM0_WIDTH          6
#define ADC_CLM0_CLM0(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM0_CLM0_SHIFT))&ADC_CLM0_CLM0_MASK)

*/

```

```

* @}

/* /* end of group ADC_Register_Masks */

/* ADC - Peripheral instance base addresses */

/** Peripheral ADC0 base address */

#define ADC0_BASE          (0x4003B000u)

/** Peripheral ADC0 base pointer */

#define ADC0          ((ADC_Type *)ADC0_BASE)

#define ADC0_BASE_PTR      (ADC0)

/** Array initializer of ADC peripheral base addresses */

#define ADC_BASE_ADDRS     { ADC0_BASE }

/** Array initializer of ADC peripheral base pointers */

#define ADC_BASE_PTRS       { ADC0 }

/*
-----  

-- ADC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros  

* @{
* /  

  

/* ADC - Register instance definitions */  

/* ADC0 */  

#define ADC0_SC1A          ADC_SC1_REG(ADC0,0)  

#define ADC0_SC1B          ADC_SC1_REG(ADC0,1)

```

#define ADC0_CFG1	ADC_CFG1_REG(ADC0)
#define ADC0_CFG2	ADC_CFG2_REG(ADC0)
#define ADC0_RA	ADC_R_REG(ADC0,0)
#define ADC0_RB	ADC_R_REG(ADC0,1)
#define ADC0_CV1	ADC_CV1_REG(ADC0)
#define ADC0_CV2	ADC_CV2_REG(ADC0)
#define ADC0_SC2	ADC_SC2_REG(ADC0)
#define ADC0_SC3	ADC_SC3_REG(ADC0)
#define ADC0_OFS	ADC_OFS_REG(ADC0)
#define ADC0_PG	ADC_PG_REG(ADC0)
#define ADC0_MG	ADC_MG_REG(ADC0)
#define ADC0_CLPD	ADC_CLPD_REG(ADC0)
#define ADC0_CLPS	ADC_CLPS_REG(ADC0)
#define ADC0_CLP4	ADC_CLP4_REG(ADC0)
#define ADC0_CLP3	ADC_CLP3_REG(ADC0)
#define ADC0_CLP2	ADC_CLP2_REG(ADC0)
#define ADC0_CLP1	ADC_CLP1_REG(ADC0)
#define ADC0_CLP0	ADC_CLP0_REG(ADC0)
#define ADC0_CLMD	ADC_CLMD_REG(ADC0)
#define ADC0_CLMS	ADC_CLMS_REG(ADC0)
#define ADC0_CLM4	ADC_CLM4_REG(ADC0)
#define ADC0_CLM3	ADC_CLM3_REG(ADC0)
#define ADC0_CLM2	ADC_CLM2_REG(ADC0)
#define ADC0_CLM1	ADC_CLM1_REG(ADC0)
#define ADC0_CLM0	ADC_CLM0_REG(ADC0)

/* ADC - Register array accessors */

#define ADC0_SC1(index)	ADC_SC1_REG(ADC0,index)
#define ADC0_R(index)	ADC_R_REG(ADC0,index)

```
/*!
 * @}
 */
/* /* end of group ADC_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* /* end of group ADC_Peripheral_Access_Layer */
```

```
/*
-----  
-- CMP Peripheral Access Layer  
----- */
```

```
/*!
 * @addtogroup CMP_Peripheral_Access_Layer CMP Peripheral Access Layer
 * @{
 */

```

```
/** CMP - Register Layout Typedef */
typedef struct {

    __IO uint8_t CRO;           /**< CMP Control Register 0, offset: 0x0 */
    __IO uint8_t CR1;           /**< CMP Control Register 1, offset: 0x1 */
    __IO uint8_t FPR;           /**< CMP Filter Period Register, offset: 0x2 */
    __IO uint8_t SCR;           /**< CMP Status and Control Register, offset: 0x3 */
    __IO uint8_t DACCR;         /**< DAC Control Register, offset: 0x4 */
    __IO uint8_t MUXCR;         /**< MUX Control Register, offset: 0x5 */

} CMP_Type, *CMP_MemMapPtr;
```

```
/*
-----  
-- CMP - Register accessor macros  
----- */  
  
/*!  
 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros  
 * {@  
 * /  
  
/* CMP - Register accessors */  
#define CMP_CR0_REG(base) ((base)->CR0)  
#define CMP_CR1_REG(base) ((base)->CR1)  
#define CMP_FPR_REG(base) ((base)->FPR)  
#define CMP_SCR_REG(base) ((base)->SCR)  
#define CMP_DACCR_REG(base) ((base)->DACCR)  
#define CMP_MUXCR_REG(base) ((base)->MUXCR)  
  
/*!  
 * @}  
 */ /* end of group CMP_Register_Accessor_Macros */
```

```
/*
-----  
-- CMP Register Masks  
----- */  
  
/*!
```

```

* @addtogroup CMP_Register_Masks CMP Register Masks

* @{
*

/* CRO Bit Fields */

#define CMP_CRO_HYSTCTR_MASK          0x3u
#define CMP_CRO_HYSTCTR_SHIFT         0
#define CMP_CRO_HYSTCTR_WIDTH         2
#define CMP_CRO_HYSTCTR(x)            (((uint8_t)((uint8_t)(x))<<CMP_CRO_HYSTCTR_SHIFT))&CMP_CRO_HYSTCTR_MASK)

#define CMP_CRO_FILTER_CNT_MASK        0x70u
#define CMP_CRO_FILTER_CNT_SHIFT       4
#define CMP_CRO_FILTER_CNT_WIDTH       3
#define CMP_CRO_FILTER_CNT(x)          (((uint8_t)((uint8_t)(x))<<CMP_CRO_FILTER_CNT_SHIFT))&CMP_CRO_FILTER_CNT_MASK)

/* CR1 Bit Fields */

#define CMP_CR1_EN_MASK               0x1u
#define CMP_CR1_EN_SHIFT              0
#define CMP_CR1_EN_WIDTH              1
#define CMP_CR1_EN(x)                (((uint8_t)((uint8_t)(x))<<CMP_CR1_EN_SHIFT))&CMP_CR1_EN_MASK)

#define CMP_CR1_OPE_MASK              0x2u
#define CMP_CR1_OPE_SHIFT             1
#define CMP_CR1_OPE_WIDTH             1
#define CMP_CR1_OPE(x)                (((uint8_t)((uint8_t)(x))<<CMP_CR1_OPE_SHIFT))&CMP_CR1_OPE_MASK)

#define CMP_CR1_COS_MASK              0x4u
#define CMP_CR1_COS_SHIFT             2
#define CMP_CR1_COS_WIDTH             1

```

```

#define CMP_CR1_COS(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_COS_SHIFT))&CMP_CR1_COS_MASK)

#define CMP_CR1_INV_MASK          0x8u
#define CMP_CR1_INV_SHIFT         3
#define CMP_CR1_INV_WIDTH         1

#define CMP_CR1_INV(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_INV_SHIFT))&CMP_CR1_INV_MASK)

#define CMP_CR1_PMODE_MASK        0x10u
#define CMP_CR1_PMODE_SHIFT       4
#define CMP_CR1_PMODE_WIDTH       1

#define CMP_CR1_PMODE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_PMODE_SHIFT))&CMP_CR1_PMODE_MASK)

#define CMP_CR1_TRIGM_MASK        0x20u
#define CMP_CR1_TRIGM_SHIFT       5
#define CMP_CR1_TRIGM_WIDTH       1

#define CMP_CR1_TRIGM(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_TRIGM_SHIFT))&CMP_CR1_TRIGM_MASK)

#define CMP_CR1_WE_MASK           0x40u
#define CMP_CR1_WE_SHIFT          6
#define CMP_CR1_WE_WIDTH          1

#define CMP_CR1_WE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_WE_SHIFT))&CMP_CR1_WE_MASK)

#define CMP_CR1_SE_MASK           0x80u
#define CMP_CR1_SE_SHIFT          7
#define CMP_CR1_SE_WIDTH          1

#define CMP_CR1_SE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_SE_SHIFT))&CMP_CR1_SE_MASK)

/* FPR Bit Fields */

#define CMP_FPR_filt_per_MASK     0xFFu
#define CMP_FPR_filt_per_SHIFT    0
#define CMP_FPR_filt_per_WIDTH    8

```

```

#define CMP_FPR_FILT_PER(x)
(((uint8_t)((uint8_t)(x)<<CMP_FPR_FILT_PER_SHIFT))&CMP_FPR_FILT_PER_MASK)

/* SCR Bit Fields */

#define CMP_SCR_COUT_MASK          0x1u
#define CMP_SCR_COUT_SHIFT         0
#define CMP_SCR_COUT_WIDTH         1

#define CMP_SCR_COUT(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_COUT_SHIFT))&CMP_SCR_COUT_MASK)

#define CMP_SCR_CFF_MASK           0x2u
#define CMP_SCR_CFF_SHIFT          1
#define CMP_SCR_CFF_WIDTH          1

#define CMP_SCR_CFF(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_CFF_SHIFT))&CMP_SCR_CFF_MASK)

#define CMP_SCR_CFR_MASK           0x4u
#define CMP_SCR_CFR_SHIFT          2
#define CMP_SCR_CFR_WIDTH          1

#define CMP_SCR_CFR(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_CFR_SHIFT))&CMP_SCR_CFR_MASK)

#define CMP_SCR_IEF_MASK           0x8u
#define CMP_SCR_IEF_SHIFT          3
#define CMP_SCR_IEF_WIDTH          1

#define CMP_SCR_IEF(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_IEF_SHIFT))&CMP_SCR_IEF_MASK)

#define CMP_SCR_IER_MASK           0x10u
#define CMP_SCR_IER_SHIFT          4
#define CMP_SCR_IER_WIDTH          1

#define CMP_SCR_IER(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_IER_SHIFT))&CMP_SCR_IER_MASK)

#define CMP_SCR_DMAEN_MASK         0x40u
#define CMP_SCR_DMAEN_SHIFT        6
#define CMP_SCR_DMAEN_WIDTH        1

```

```

#define CMP_SCR_DMAEN(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_DMAEN_SHIFT))&CMP_SCR_DMAEN_MASK)

/* DACC Bit Fields */

#define CMP_DACCR_VOSEL_MASK          0x3Fu
#define CMP_DACCR_VOSEL_SHIFT         0
#define CMP_DACCR_VOSEL_WIDTH         6
#define CMP_DACCR_VOSEL(x)
(((uint8_t)((uint8_t)(x)<<CMP_DACCR_VOSEL_SHIFT))&CMP_DACCR_VOSEL_MASK)

#define CMP_DACCR_VRSEL_MASK          0x40u
#define CMP_DACCR_VRSEL_SHIFT         6
#define CMP_DACCR_VRSEL_WIDTH         1
#define CMP_DACCR_VRSEL(x)
(((uint8_t)((uint8_t)(x)<<CMP_DACCR_VRSEL_SHIFT))&CMP_DACCR_VRSEL_MASK)

#define CMP_DACCR_DACEN_MASK          0x80u
#define CMP_DACCR_DACEN_SHIFT         7
#define CMP_DACCR_DACEN_WIDTH         1
#define CMP_DACCR_DACEN(x)
(((uint8_t)((uint8_t)(x)<<CMP_DACCR_DACEN_SHIFT))&CMP_DACCR_DACEN_MASK)

/* MUXCR Bit Fields */

#define CMP_MUXCR_MSEL_MASK           0x7u
#define CMP_MUXCR_MSEL_SHIFT          0
#define CMP_MUXCR_MSEL_WIDTH          3
#define CMP_MUXCR_MSEL(x)
(((uint8_t)((uint8_t)(x)<<CMP_MUXCR_MSEL_SHIFT))&CMP_MUXCR_MSEL_MASK)

#define CMP_MUXCR_PSEL_MASK           0x38u
#define CMP_MUXCR_PSEL_SHIFT          3
#define CMP_MUXCR_PSEL_WIDTH          3
#define CMP_MUXCR_PSEL(x)
(((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSEL_SHIFT))&CMP_MUXCR_PSEL_MASK)

#define CMP_MUXCR_PSTM_MASK           0x80u
#define CMP_MUXCR_PSTM_SHIFT          7

```

```

#define CMP_MUXCR_PSTM_WIDTH          1

#define CMP_MUXCR_PSTM(x) \
(((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSTM_SHIFT))&CMP_MUXCR_PSTM_MASK)

/*
 * @}
 */

/* /* end of group CMP_Register_Masks */

/* CMP - Peripheral instance base addresses */

/** Peripheral CMPO base address */

#define CMPO_BASE           (0x40073000u)

/** Peripheral CMPO base pointer */

#define CMPO                ((CMP_Type *)CMPO_BASE)

#define CMPO_BASE_PTR        (CMPO)

/** Array initializer of CMP peripheral base addresses */

#define CMP_BASE_ADDRS      { CMPO_BASE }

/** Array initializer of CMP peripheral base pointers */

#define CMP_BASE_PTRS        { CMPO }

/*
-----*
-- CMP - Register accessor macros
----- */

```

```

/*
 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros
 * @{
 */


```

```
/* CMP - Register instance definitions */

/* CMPO */

#define CMPO_CRO           CMP_CRO_REG(CMPO)
#define CMPO_CR1           CMP_CR1_REG(CMPO)
#define CMPO_FPR           CMP_FPR_REG(CMPO)
#define CMPO_SCR           CMP_SCR_REG(CMPO)
#define CMPO_DACCR         CMP_DACCR_REG(CMPO)
#define CMPO_MUXCR         CMP_MUXCR_REG(CMPO)
```

```
/*
 * @}
 */
/* end of group CMP_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group CMP_Peripheral_Access_Layer */
```

```
/*
-----  
-- DAC Peripheral Access Layer  
----- */
```

```
/*
* @addtogroup DAC_Peripheral_Access_Layer DAC Peripheral Access Layer
* @{
* /
```

```

/** DAC - Register Layout Typedef */

typedef struct {

    struct {                                     /* offset: 0x0, array step: 0x2 */
        __IO uint8_t DATL;                      /* **< DAC Data Low Register, array offset: 0x0, array step: 0x2 */
        __IO uint8_t DATH;                      /* **< DAC Data High Register, array offset: 0x1, array step: 0x2 */
    } DAT[2];

    uint8_t RESERVED_0[28];

    __IO uint8_t SR;                          /* **< DAC Status Register, offset: 0x20 */
    __IO uint8_t C0;                          /* **< DAC Control Register, offset: 0x21 */
    __IO uint8_t C1;                          /* **< DAC Control Register 1, offset: 0x22 */
    __IO uint8_t C2;                          /* **< DAC Control Register 2, offset: 0x23 */

} DAC_Type, *DAC_MemMapPtr;

```

```

/*
-----  

-- DAC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros  

* @{
* /  


```

```

/* DAC - Register accessors */  

#define DAC_DATL_REG(base,index)      ((base)->DAT[index].DATL)  

#define DAC_DATL_COUNT                2  

#define DAC_DATH_REG(base,index)      ((base)->DAT[index].DATH)  

#define DAC_DATH_COUNT                2  

#define DAC_SR_REG(base)              ((base)->SR)

```

```
#define DAC_C0_REG(base) ((base)->C0)
#define DAC_C1_REG(base) ((base)->C1)
#define DAC_C2_REG(base) ((base)->C2)

/*!
 * @}
 */
/* /* end of group DAC_Register_Accessor_Macros */
```

```
/*
-----  
-- DAC Register Masks  
----- */
```

```
/*!  
 * @addtogroup DAC_Register_Masks DAC Register Masks  
 * @{  
 */
```

```
/* DATL Bit Fields */  
  
#define DAC_DATL_DATA0_MASK 0xFFu  
#define DAC_DATL_DATA0_SHIFT 0  
#define DAC_DATL_DATA0_WIDTH 8  
  
#define DAC_DATL_DATA0(x)  
(((uint8_t)((uint8_t)(x))<<DAC_DATL_DATA0_SHIFT))&DAC_DATL_DATA0_MASK)  
  
/* DATH Bit Fields */  
  
#define DAC_DATH_DATA1_MASK 0xFu  
#define DAC_DATH_DATA1_SHIFT 0  
#define DAC_DATH_DATA1_WIDTH 4
```

```

#define DAC_DATH_DATA1(x)
(((uint8_t)((uint8_t)(x)<<DAC_DATH_DATA1_SHIFT))&DAC_DATH_DATA1_MASK)

/* SR Bit Fields */

#define DAC_SR_DACBFRPBF_MASK          0x1u
#define DAC_SR_DACBFRPBF_SHIFT         0
#define DAC_SR_DACBFRPBF_WIDTH         1
#define DAC_SR_DACBFRPBF(x)
(((uint8_t)((uint8_t)(x)<<DAC_SR_DACBFRPBF_SHIFT))&DAC_SR_DACBFRPBF_MASK)

#define DAC_SR_DACBFRPTF_MASK          0x2u
#define DAC_SR_DACBFRPTF_SHIFT         1
#define DAC_SR_DACBFRPTF_WIDTH         1
#define DAC_SR_DACBFRPTF(x)
(((uint8_t)((uint8_t)(x)<<DAC_SR_DACBFRPTF_SHIFT))&DAC_SR_DACBFRPTF_MASK)

/* CO Bit Fields */

#define DAC_C0_DACBBIEN_MASK           0x1u
#define DAC_C0_DACBBIEN_SHIFT          0
#define DAC_C0_DACBBIEN_WIDTH          1
#define DAC_C0_DACBBIEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C0_DACBBIEN_SHIFT))&DAC_C0_DACBBIEN_MASK)

#define DAC_C0_DACBTIEN_MASK           0x2u
#define DAC_C0_DACBTIEN_SHIFT          1
#define DAC_C0_DACBTIEN_WIDTH          1
#define DAC_C0_DACBTIEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C0_DACBTIEN_SHIFT))&DAC_C0_DACBTIEN_MASK)

#define DAC_C0_LPEN_MASK               0x8u
#define DAC_C0_LPEN_SHIFT              3
#define DAC_C0_LPEN_WIDTH              1
#define DAC_C0_LPEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C0_LPEN_SHIFT))&DAC_C0_LPEN_MASK)

#define DAC_C0_DACSWTRG_MASK           0x10u
#define DAC_C0_DACSWTRG_SHIFT          4

```

```

#define DAC_C0_DACSWTRG_WIDTH          1
#define DAC_C0_DACSWTRG(x)             (((uint8_t)((uint8_t)(x)<<DAC_C0_DACSWTRG_SHIFT))&DAC_C0_DACSWTRG_MASK)
#define DAC_C0_DACTRGSEL_MASK          0x20u
#define DAC_C0_DACTRGSEL_SHIFT         5
#define DAC_C0_DACTRGSEL_WIDTH         1
#define DAC_C0_DACTRGSEL(x)            (((uint8_t)((uint8_t)(x)<<DAC_C0_DACTRGSEL_SHIFT))&DAC_C0_DACTRGSEL_MASK)
#define DAC_C0_DACRFS_MASK             0x40u
#define DAC_C0_DACRFS_SHIFT            6
#define DAC_C0_DACRFS_WIDTH            1
#define DAC_C0_DACRFS(x)               (((uint8_t)((uint8_t)(x)<<DAC_C0_DACRFS_SHIFT))&DAC_C0_DACRFS_MASK)
#define DAC_C0_DACEN_MASK              0x80u
#define DAC_C0_DACEN_SHIFT             7
#define DAC_C0_DACEN_WIDTH             1
#define DAC_C0_DACEN(x)                (((uint8_t)((uint8_t)(x)<<DAC_C0_DACEN_SHIFT))&DAC_C0_DACEN_MASK)
/* C1 Bit Fields */
#define DAC_C1_DACBFEN_MASK            0x1u
#define DAC_C1_DACBFEN_SHIFT           0
#define DAC_C1_DACBFEN_WIDTH           1
#define DAC_C1_DACBFEN(x)              (((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFEN_SHIFT))&DAC_C1_DACBFEN_MASK)
#define DAC_C1_DACBFMD_MASK            0x4u
#define DAC_C1_DACBFMD_SHIFT           2
#define DAC_C1_DACBFMD_WIDTH           1
#define DAC_C1_DACBFMD(x)              (((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFMD_SHIFT))&DAC_C1_DACBFMD_MASK)
#define DAC_C1_DMAEN_MASK              0x80u
#define DAC_C1_DMAEN_SHIFT             7

```

```

#define DAC_C1_DMAEN_WIDTH          1
#define DAC_C1_DMAEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C1_DMAEN_SHIFT))&DAC_C1_DMAEN_MASK)
/* C2 Bit Fields */
#define DAC_C2_DACBFUP_MASK         0x1u
#define DAC_C2_DACBFUP_SHIFT        0
#define DAC_C2_DACBFUP_WIDTH        1
#define DAC_C2_DACBFUP(x) (((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFUP_SHIFT))&DAC_C2_DACBFUP_MASK)
#define DAC_C2_DACBFRP_MASK         0x10u
#define DAC_C2_DACBFRP_SHIFT        4
#define DAC_C2_DACBFRP_WIDTH        1
#define DAC_C2_DACBFRP(x) (((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFRP_SHIFT))&DAC_C2_DACBFRP_MASK)

/*
 * @}
 */

/* /* end of group DAC_Register_Masks */

```

```

/* DAC - Peripheral instance base addresses */
/** Peripheral DAC0 base address */
#define DAC0_BASE                  (0x4003F000u)
/** Peripheral DAC0 base pointer */
#define DAC0                      ((DAC_Type *)DAC0_BASE)
#define DAC0_BASE_PTR              (DAC0)
/** Array initializer of DAC peripheral base addresses */
#define DAC_BASE_ADDRS             { DAC0_BASE }
/** Array initializer of DAC peripheral base pointers */
#define DAC_BASE_PTRS              { DAC0 }

```

```

/*
-----  

-- DAC - Register accessor macros  

----- */
  

/*!  

 * @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros  

 * {@  

 */  

  

/* DAC - Register instance definitions */  

/* DAC0 */  

#define DAC0_DAT0L          DAC_DATL_REG(DAC0,0)  

#define DAC0_DAT0H          DAC_DATH_REG(DAC0,0)  

#define DAC0_DAT1L          DAC_DATL_REG(DAC0,1)  

#define DAC0_DAT1H          DAC_DATH_REG(DAC0,1)  

#define DAC0_SR              DAC_SR_REG(DAC0)  

#define DAC0_C0              DAC_C0_REG(DAC0)  

#define DAC0_C1              DAC_C1_REG(DAC0)  

#define DAC0_C2              DAC_C2_REG(DAC0)  

  

/* DAC - Register array accessors */  

#define DAC0_DATL(index)     DAC_DATL_REG(DAC0,index)  

#define DAC0_DATH(index)     DAC_DATH_REG(DAC0,index)  

  

/*!  

 * @}  

 */ /* end of group DAC_Register_Accessor_Macros */

```

```
/*!
 * @}
 */ /* end of group DAC_Peripheral_Access_Layer */

/*
-----  
-- DMA Peripheral Access Layer  
----- */

```

```
/*!  
 * @addtogroup DMA_Peripheral_Access_Layer DMA Peripheral Access Layer  
 * @{  
 */  
  
/** DMA - Register Layout Typedef */  
  
typedef struct {  
    uint8_t RESERVED_0[256];  
  
    struct {  
        /* offset: 0x100, array step: 0x10 */  
        __IO uint32_t SAR;           /* **< Source Address Register, array offset: 0x100, array step:  
        0x10 */  
  
        __IO uint32_t DAR;           /* **< Destination Address Register, array offset: 0x104, array step:  
        0x10 */  
  
        union {  
            /* offset: 0x108, array step: 0x10 */  
            __IO uint32_t DSR_BCR;     /* **< DMA Status Register / Byte Count Register, array offset:  
            0x108, array step: 0x10 */  
  
            struct {  
                /* offset: 0x108, array step: 0x10 */  
                uint8_t RESERVED_0[3];  
            };  
        };  
    };  
};
```

```

    __IO uint8_t DSR;           /**< DMA_DSR0 register...DMA_DSR3 register., array offset:
0x10B, array step: 0x10 */
} DMA_DSR_ACCESS8BIT;

};

__IO uint32_t DCR;           /**< DMA Control Register, array offset: 0x10C, array step: 0x10
*/
} DMA[4];

} DMA_Type, *DMA_MemMapPtr;

```

```

/*
-----  

-- DMA - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros  

* @{
* /

```

```

/* DMA - Register accessors */  

#define DMA_SAR_REG(base,index)      ((base)->DMA[index].SAR)  

#define DMA_SAR_COUNT                4  

#define DMA_DAR_REG(base,index)      ((base)->DMA[index].DAR)  

#define DMA_DAR_COUNT                4  

#define DMA_DSR_BCR_REG(base,index)   ((base)->DMA[index].DSR_BCR)  

#define DMA_DSR_BCR_COUNT            4  

#define DMA_DSR_REG(base,index)      ((base)->DMA[index].DMA_DSR_ACCESS8BIT.DSR)  

#define DMA_DSR_COUNT                4  

#define DMA_DCR_REG(base,index)      ((base)->DMA[index].DCR)

```

```
#define DMA_DCR_COUNT          4

/*!
 * @}
 */ /* end of group DMA_Register_Accessor_Macros */
```

```
/*
-----  
-- DMA Register Masks  
----- */
```

```
/*!  
 * @addtogroup DMA_Register_Masks DMA Register Masks  
 * @{  
 */
```

```
/* SAR Bit Fields */

#define DMA_SAR_SAR_MASK          0xFFFFFFFFu
#define DMA_SAR_SAR_SHIFT         0
#define DMA_SAR_SAR_WIDTH         32
#define DMA_SAR_SAR(x)            (((uint32_t)((uint32_t)(x))<<DMA_SAR_SAR_SHIFT))&DMA_SAR_SAR_MASK

/* DAR Bit Fields */

#define DMA_DAR_DAR_MASK          0xFFFFFFFFu
#define DMA_DAR_DAR_SHIFT         0
#define DMA_DAR_DAR_WIDTH         32
#define DMA_DAR_DAR(x)            (((uint32_t)((uint32_t)(x))<<DMA_DAR_DAR_SHIFT))&DMA_DAR_DAR_MASK

/* DSR_BCR Bit Fields */
```

```

#define DMA_DSR_BCR_BCR_MASK          0xFFFFFFFFu
#define DMA_DSR_BCR_BCR_SHIFT         0
#define DMA_DSR_BCR_BCR_WIDTH         24
#define DMA_DSR_BCR_BCR(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BCR_SHIFT))&DMA_DSR_BCR_BCR_MASK)

#define DMA_DSR_BCR_DONE_MASK         0x1000000u
#define DMA_DSR_BCR_DONE_SHIFT        24
#define DMA_DSR_BCR_DONE_WIDTH        1
#define DMA_DSR_BCR_DONE(x)           (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_DONE_SHIFT))&DMA_DSR_BCR_DONE_MASK)

#define DMA_DSR_BCR_BSY_MASK          0x2000000u
#define DMA_DSR_BCR_BSY_SHIFT         25
#define DMA_DSR_BCR_BSY_WIDTH         1
#define DMA_DSR_BCR_BSY(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BSY_SHIFT))&DMA_DSR_BCR_BSY_MASK)

#define DMA_DSR_BCR_REQ_MASK          0x4000000u
#define DMA_DSR_BCR_REQ_SHIFT         26
#define DMA_DSR_BCR_REQ_WIDTH         1
#define DMA_DSR_BCR_REQ(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_REQ_SHIFT))&DMA_DSR_BCR_REQ_MASK)

#define DMA_DSR_BCR_BED_MASK          0x10000000u
#define DMA_DSR_BCR_BED_SHIFT         28
#define DMA_DSR_BCR_BED_WIDTH         1
#define DMA_DSR_BCR_BED(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BED_SHIFT))&DMA_DSR_BCR_BED_MASK)

#define DMA_DSR_BCR_BES_MASK          0x20000000u
#define DMA_DSR_BCR_BES_SHIFT         29
#define DMA_DSR_BCR_BES_WIDTH         1
#define DMA_DSR_BCR_BES(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BES_SHIFT))&DMA_DSR_BCR_BES_MASK)

#define DMA_DSR_BCR_CE_MASK          0x40000000u

```

```

#define DMA_DSR_BCR_CE_SHIFT          30
#define DMA_DSR_BCR_CE_WIDTH          1
#define DMA_DSR_BCR_CE(x) (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_CE_SHIFT))&DMA_DSR_BCR_CE_MASK)

/* DCR Bit Fields */

#define DMA_DCR_LCH2_MASK            0x3u
#define DMA_DCR_LCH2_SHIFT           0
#define DMA_DCR_LCH2_WIDTH           2
#define DMA_DCR_LCH2(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH2_SHIFT))&DMA_DCR_LCH2_MASK)

#define DMA_DCR_LCH1_MASK            0xCu
#define DMA_DCR_LCH1_SHIFT           2
#define DMA_DCR_LCH1_WIDTH           2
#define DMA_DCR_LCH1(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH1_SHIFT))&DMA_DCR_LCH1_MASK)

#define DMA_DCR_LINKCC_MASK          0x30u
#define DMA_DCR_LINKCC_SHIFT         4
#define DMA_DCR_LINKCC_WIDTH         2
#define DMA_DCR_LINKCC(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_LINKCC_SHIFT))&DMA_DCR_LINKCC_MASK)

#define DMA_DCR_D_REQ_MASK           0x80u
#define DMA_DCR_D_REQ_SHIFT          7
#define DMA_DCR_D_REQ_WIDTH          1
#define DMA_DCR_D_REQ(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_D_REQ_SHIFT))&DMA_DCR_D_REQ_MASK)

#define DMA_DCR_DMOD_MASK            0xF00u
#define DMA_DCR_DMOD_SHIFT           8
#define DMA_DCR_DMOD_WIDTH           4
#define DMA_DCR_DMOD(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DMOD_SHIFT))&DMA_DCR_DMOD_MASK)

#define DMA_DCR_SMOD_MASK            0xF000u

```

```

#define DMA_DCR_SMOD_SHIFT          12
#define DMA_DCR_SMOD_WIDTH          4
#define DMA_DCR_SMOD(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_SMOD_SHIFT))&DMA_DCR_SMOD_MASK)

#define DMA_DCR_START_MASK          0x10000u
#define DMA_DCR_START_SHIFT          16
#define DMA_DCR_START_WIDTH          1
#define DMA_DCR_START(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_START_SHIFT))&DMA_DCR_START_MASK)

#define DMA_DCR_DSIZEMASK          0x60000u
#define DMA_DCR_DSIZE_SHIFT          17
#define DMA_DCR_DSIZE_WIDTH          2
#define DMA_DCR_DSIZE(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DSIZE_SHIFT))&DMA_DCR_DSIZE_MASK)

#define DMA_DCR_DINC_MASK           0x80000u
#define DMA_DCR_DINC_SHIFT           19
#define DMA_DCR_DINC_WIDTH           1
#define DMA_DCR_DINC(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_DINC_SHIFT))&DMA_DCR_DINC_MASK)

#define DMA_DCR_SSIZEMASK           0x300000u
#define DMA_DCR_SSIZE_SHIFT           20
#define DMA_DCR_SSIZE_WIDTH           2
#define DMA_DCR_SSIZE(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_SSIZE_SHIFT))&DMA_DCR_SSIZE_MASK)

#define DMA_DCR_SINC_MASK           0x400000u
#define DMA_DCR_SINC_SHIFT           22
#define DMA_DCR_SINC_WIDTH           1
#define DMA_DCR_SINC(x) (((uint32_t)((uint32_t)(x)<<DMA_DCR_SINC_SHIFT))&DMA_DCR_SINC_MASK)

#define DMA_DCR_EADREQ_MASK          0x800000u
#define DMA_DCR_EADREQ_SHIFT          23

```

```

#define DMA_DCR_EADREQ_WIDTH          1
#define DMA_DCR_EADREQ(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_EADREQ_SHIFT))&DMA_DCR_EADREQ_MASK)
#define DMA_DCR_AA_MASK               0x10000000u
#define DMA_DCR_AA_SHIFT              28
#define DMA_DCR_AA_WIDTH              1
#define DMA_DCR_AA(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_AA_SHIFT))&DMA_DCR_AA_MASK)
#define DMA_DCR_CS_MASK               0x20000000u
#define DMA_DCR_CS_SHIFT              29
#define DMA_DCR_CS_WIDTH              1
#define DMA_DCR_CS(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_CS_SHIFT))&DMA_DCR_CS_MASK)
#define DMA_DCR_ERQ_MASK              0x40000000u
#define DMA_DCR_ERQ_SHIFT             30
#define DMA_DCR_ERQ_WIDTH             1
#define DMA_DCR_ERQ(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_ERQ_SHIFT))&DMA_DCR_ERQ_MASK)
#define DMA_DCR_EINT_MASK              0x80000000u
#define DMA_DCR_EINT_SHIFT             31
#define DMA_DCR_EINT_WIDTH             1
#define DMA_DCR_EINT(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_EINT_SHIFT))&DMA_DCR_EINT_MASK)

/*
 * @}
 */

/* /* end of group DMA_Register_Masks */

/* DMA - Peripheral instance base addresses */

```

```

/** Peripheral DMA base address */
#define DMA_BASE           (0x40008000u)

/** Peripheral DMA base pointer */
#define DMA0                ((DMA_Type *)DMA_BASE)

#define DMA_BASE_PTR         (DMA0)

/** Array initializer of DMA peripheral base addresses */
#define DMA_BASE_ADDRS      { DMA_BASE }

/** Array initializer of DMA peripheral base pointers */
#define DMA_BASE_PTRS        { DMA0 }

```

```

/*
-----*
-- DMA - Register accessor macros
----- */

```

```

/*!

```

```

* @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros
* @{
* /

```

```

/* DMA - Register instance definitions */

/* DMA */

#define DMA_SAR0            DMA_SAR_REG(DMA0,0)
#define DMA_DAR0            DMA_DAR_REG(DMA0,0)
#define DMA_DSR_BCR0        DMA_DSR_BCR_REG(DMA0,0)
#define DMA_DSR0             DMA_DSR_REG(DMA0,0)
#define DMA_DCR0             DMA_DCR_REG(DMA0,0)
#define DMA_SAR1            DMA_SAR_REG(DMA0,1)
#define DMA_DAR1            DMA_DAR_REG(DMA0,1)

```

```

#define DMA_DSR_BCR1           DMA_DSR_BCR_REG(DMA0,1)
#define DMA_DSR1                DMA_DSR_REG(DMA0,1)
#define DMA_DCR1                DMA_DCR_REG(DMA0,1)
#define DMA_SAR2                DMA_SAR_REG(DMA0,2)
#define DMA_DAR2                DMA_DAR_REG(DMA0,2)
#define DMA_DSR_BCR2           DMA_DSR_BCR_REG(DMA0,2)
#define DMA_DSR2                DMA_DSR_REG(DMA0,2)
#define DMA_DCR2                DMA_DCR_REG(DMA0,2)
#define DMA_SAR3                DMA_SAR_REG(DMA0,3)
#define DMA_DAR3                DMA_DAR_REG(DMA0,3)
#define DMA_DSR_BCR3           DMA_DSR_BCR_REG(DMA0,3)
#define DMA_DSR3                DMA_DSR_REG(DMA0,3)
#define DMA_DCR3                DMA_DCR_REG(DMA0,3)

```

```

/* DMA - Register array accessors */

#define DMA_SAR(index)          DMA_SAR_REG(DMA0,index)
#define DMA_DAR(index)          DMA_DAR_REG(DMA0,index)
#define DMA_DSR_BCR(index)      DMA_DSR_BCR_REG(DMA0,index)
#define DMA_DSR(index)          DMA_DSR_REG(DMA0,index)
#define DMA_DCR(index)          DMA_DCR_REG(DMA0,index)

```

```

!*/
* @}

/* end of group DMA_Register_Accessor_Macros */

```

```

!*
* @}

/* end of group DMA_Peripheral_Access_Layer */

```

```

/*
-----  

-- DMAMUX Peripheral Access Layer  

----- */
  

/*!  

 * @addtogroup DMAMUX_Peripheral_Access_Layer DMAMUX Peripheral Access Layer
 * @{
 * /  

  

/** DMAMUX - Register Layout Typedef */  

typedef struct {  

    __IO uint8_t CHCFG[4];           /**< Channel Configuration register, array offset: 0x0, array step:  

    0x1 */  

} DMAMUX_Type, *DMAMUX_MemMapPtr;  

  

/*
-----  

-- DMAMUX - Register accessor macros  

----- */
  

/*!  

 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros
 * @{
 * /  

  

/* DMAMUX - Register accessors */  

#define DMAMUX_CHCFG_REG(base,index) ((base)->CHCFG[index])

```

```

#define DMAMUX_CHCFG_COUNT          4

/*
 * @}
 */

/* /* end of group DMAMUX_Register_Accessor_Macros */

/*
-----*
-- DMAMUX Register Masks
----- */
/*!

 * @addtogroup DMAMUX_Register_Masks DMAMUX Register Masks
 * @{
 */

/* CHCFG Bit Fields */

#define DMAMUX_CHCFG_SOURCE_MASK      0x3Fu
#define DMAMUX_CHCFG_SOURCE_SHIFT     0
#define DMAMUX_CHCFG_SOURCE_WIDTH     6
#define DMAMUX_CHCFG_SOURCE(x)        (((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_SOURCE_SHIFT))&DMAMUX_CHCFG_SOURCE_MASK)

#define DMAMUX_CHCFG_TRIG_MASK        0x40u
#define DMAMUX_CHCFG_TRIG_SHIFT       6
#define DMAMUX_CHCFG_TRIG_WIDTH       1
#define DMAMUX_CHCFG_TRIG(x)          (((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_TRIG_SHIFT))&DMAMUX_CHCFG_TRIG_MASK)

#define DMAMUX_CHCFG_ENBL_MASK        0x80u
#define DMAMUX_CHCFG_ENBL_SHIFT       7

```

```

#define DMAMUX_CHCFG_ENBL_WIDTH          1

#define DMAMUX_CHCFG_ENBL(x)
(((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_ENBL_SHIFT))&DMAMUX_CHCFG_ENBL_MASK)

/*
 * @}
 */

/* /* end of group DMAMUX_Register_Masks */

/* DMAMUX - Peripheral instance base addresses */

/** Peripheral DMAMUX0 base address */
#define DMAMUX0_BASE          (0x40021000u)

/** Peripheral DMAMUX0 base pointer */
#define DMAMUX0          ((DMAMUX_Type *)DMAMUX0_BASE)

#define DMAMUX0_BASE_PTR      (DMAMUX0)

/** Array initializer of DMAMUX peripheral base addresses */
#define DMAMUX_BASE_ADDRS     { DMAMUX0_BASE }

/** Array initializer of DMAMUX peripheral base pointers */
#define DMAMUX_BASE_PTRS      { DMAMUX0 }

/*
-----*
-- DMAMUX - Register accessor macros
----- */

```

```

/*
 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros
 * @{
 */


```

```

/* DMAMUX - Register instance definitions */

/* DMAMUX0 */

#define DMAMUX0_CHCFG0          DMAMUX_CHCFG_REG(DMAMUX0,0)
#define DMAMUX0_CHCFG1          DMAMUX_CHCFG_REG(DMAMUX0,1)
#define DMAMUX0_CHCFG2          DMAMUX_CHCFG_REG(DMAMUX0,2)
#define DMAMUX0_CHCFG3          DMAMUX_CHCFG_REG(DMAMUX0,3)

/* DMAMUX - Register array accessors */

#define DMAMUX0_CHCFG(index)     DMAMUX_CHCFG_REG(DMAMUX0,index)

/*
 * @{
 */
/* /* end of group DMAMUX_Register_Accessor_Macros */

/*
 * @{
*/
/* /* end of group DMAMUX_Peripheral_Access_Layer */

/*
-----*
-- FGPIO Peripheral Access Layer
----- */
/*!

* @addtogroup FGPIO_Peripheral_Access_Layer FGPIO Peripheral Access Layer
* @{
*/

```

```
/** FGPIO - Register Layout Typedef */

typedef struct {

    __IO uint32_t PDOR;           /**< Port Data Output Register, offset: 0x0 */
    __O  uint32_t PSOR;          /**< Port Set Output Register, offset: 0x4 */
    __O  uint32_t PCOR;          /**< Port Clear Output Register, offset: 0x8 */
    __O  uint32_t PTOR;          /**< Port Toggle Output Register, offset: 0xC */
    __I  uint32_t PDIR;          /**< Port Data Input Register, offset: 0x10 */
    __IO uint32_t PDDR;          /**< Port Data Direction Register, offset: 0x14 */

} FGPIO_Type, *FGPIO_MemMapPtr;
```

```
/*
-- FGPIO - Register accessor macros
----- */
```

```
/*!
* @addtogroup FGPIO_Register_Accessor_Macros FGPIO - Register accessor macros
* @{
*/

```

```
/* FGPIO - Register accessors */

#define FGPIO_PDOR_REG(base)      ((base)->PDOR)
#define FGPIO_PSOR_REG(base)      ((base)->PSOR)
#define FGPIO_PCOR_REG(base)      ((base)->PCOR)
#define FGPIO_PTOR_REG(base)      ((base)->PTOR)
#define FGPIO_PDIR_REG(base)      ((base)->PDIR)
#define FGPIO_PDDR_REG(base)      ((base)->PDDR)
```

```
/*!  
 * @}  
 */ /* end of group FGPIO_Register_Accessor_Macros */
```

```
/* -----  
-- FGPIO Register Masks  
----- */
```

```
/*!  
 * @addtogroup FGPIO_Register_Masks FGPIO Register Masks  
 * @{  
 */
```

```
/* PDOR Bit Fields */
```

```
#define FGPIO_PDOR_PDO_MASK      0xFFFFFFFFu  
#define FGPIO_PDOR_PDO_SHIFT     0  
#define FGPIO_PDOR_PDO_WIDTH    32  
  
#define FGPIO_PDOR_PDO(x)  
(((uint32_t)((uint32_t)(x)<<FGPIO_PDOR_PDO_SHIFT))&FGPIO_PDOR_PDO_MASK)
```

```
/* PSOR Bit Fields */
```

```
#define FGPIO_PSOR PTSO_MASK      0xFFFFFFFFu  
#define FGPIO_PSOR PTSO_SHIFT     0  
#define FGPIO_PSOR PTSO_WIDTH    32  
  
#define FGPIO_PSOR PTSO(x)  
(((uint32_t)((uint32_t)(x)<<FGPIO_PSOR PTSO_SHIFT))&FGPIO_PSOR PTSO_MASK)
```

```
/* PCOR Bit Fields */
```

```
#define FGPIO_PCOR_PTCO_MASK      0xFFFFFFFFu  
#define FGPIO_PCOR_PTCO_SHIFT     0
```

```

#define GPIO_PCOR_PTCO_WIDTH          32
#define GPIO_PCOR_PTCO(x) (((uint32_t)((uint32_t)(x)<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)
/* PTOR Bit Fields */

#define GPIO_PTOR_PTTO_MASK           0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT          0
#define GPIO_PTOR_PTTO_WIDTH          32
#define GPIO_PTOR_PTTO(x) (((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/* PDIR Bit Fields */

#define GPIO_PDIR_PDI_MASK            0xFFFFFFFFu
#define GPIO_PDIR_PDI_SHIFT           0
#define GPIO_PDIR_PDI_WIDTH           32
#define GPIO_PDIR_PDI(x) (((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK            0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT           0
#define GPIO_PDDR_PDD_WIDTH           32
#define GPIO_PDDR_PDD(x) (((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

/*!
 * @}
 */
/* /* end of group GPIO_Register_Masks */

```

```

/* GPIO - Peripheral instance base addresses */
/** Peripheral GPIOA base address */
#define GPIOA_BASE                   (0xF80FF000u)

```

```

/** Peripheral FGPIOA base pointer */

#define FGPIOA           ((GPIO_Type *)FGPIOA_BASE)

#define FGPIOA_BASE_PTR    (FGPIOA)

/** Peripheral FGPIOB base address */

#define FGPIOB_BASE        (0xF80FF040u)

/** Peripheral FGPIOB base pointer */

#define FGPIOB             ((GPIO_Type *)FGPIOB_BASE)

#define FGPIOB_BASE_PTR    (FGPIOB)

/** Peripheral FGPIOC base address */

#define FGPIOC_BASE        (0xF80FF080u)

/** Peripheral FGPIOC base pointer */

#define FGPIOC              ((GPIO_Type *)FGPIOC_BASE)

#define FGPIOC_BASE_PTR    (FGPIOC)

/** Peripheral FGPIOD base address */

#define FGPIOD_BASE         (0xF80FF0C0u)

/** Peripheral FGPIOD base pointer */

#define FGPIOD              ((GPIO_Type *)FGPIOD_BASE)

#define FGPIOD_BASE_PTR    (FGPIOD)

/** Peripheral FGPIOE base address */

#define FGPIOE_BASE          (0xF80FF100u)

/** Peripheral FGPIOE base pointer */

#define FGPIOE              ((GPIO_Type *)FGPIOE_BASE)

#define FGPIOE_BASE_PTR    (FGPIOE)

/** Array initializer of GPIO peripheral base addresses */

#define GPIO_BASE_ADDRS      { FGPIOA_BASE, FGPIOB_BASE, FGPIOC_BASE, FGPIOD_BASE,
                           FGPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */

#define GPIO_BASE_PTRS       { FGPIOA, FGPIOB, FGPIOC, FGPIOD, FGPIOE }

```

```

/*
-- FGPIO - Register accessor macros
----- */
/*!
* @addtogroup FGPIO_Register_Accessor_Macros FGPIO - Register accessor macros
* @{
*/
/* FGPIO - Register instance definitions */

/* FGPIOA */
#define FGPIOA_PDOR      FGPIO_PDOR_REG(GPIOA)
#define FGPIOA_PSOR      FGPIO_PSOR_REG(GPIOA)
#define FGPIOA_PCOR      FGPIO_PCOR_REG(GPIOA)
#define FGPIOA_PTOR      FGPIO_PTOR_REG(GPIOA)
#define FGPIOA_PDIR      FGPIO_PDIR_REG(GPIOA)
#define FGPIOA_PDDR      FGPIO_PDDR_REG(GPIOA)

/* FGPIOB */
#define FGPIOB_PDOR      FGPIO_PDOR_REG(GPIOB)
#define FGPIOB_PSOR      FGPIO_PSOR_REG(GPIOB)
#define FGPIOB_PCOR      FGPIO_PCOR_REG(GPIOB)
#define FGPIOB_PTOR      FGPIO_PTOR_REG(GPIOB)
#define FGPIOB_PDIR      FGPIO_PDIR_REG(GPIOB)
#define FGPIOB_PDDR      FGPIO_PDDR_REG(GPIOB)

/* FGPIOC */
#define FGPIOC_PDOR      FGPIO_PDOR_REG(GPIOC)
#define FGPIOC_PSOR      FGPIO_PSOR_REG(GPIOC)
#define FGPIOC_PCOR      FGPIO_PCOR_REG(GPIOC)

```

```

#define FGPIOC_PTOR           FGPIO_PTOR_REG(FGPIOC)
#define FGPIOC_PDIR           FGPIO_PDIR_REG(FGPIOC)
#define FGPIOC_PDDR           FGPIO_PDDR_REG(FGPIOC)
/* FGPIOD */
#define FGPIOD_PDOR           FGPIO_PDOR_REG(FGPIOD)
#define FGPIOD_PSOR           FGPIO_PSOR_REG(FGPIOD)
#define FGPIOD_PCOR           FGPIO_PCOR_REG(FGPIOD)
#define FGPIOD_PTOR           FGPIO_PTOR_REG(FGPIOD)
#define FGPIOD_PDIR           FGPIO_PDIR_REG(FGPIOD)
#define FGPIOD_PDDR           FGPIO_PDDR_REG(FGPIOD)
/* FGPIOE */
#define FGPIOE_PDOR           FGPIO_PDOR_REG(FGPIOE)
#define FGPIOE_PSOR           FGPIO_PSOR_REG(FGPIOE)
#define FGPIOE_PCOR           FGPIO_PCOR_REG(FGPIOE)
#define FGPIOE_PTOR           FGPIO_PTOR_REG(FGPIOE)
#define FGPIOE_PDIR           FGPIO_PDIR_REG(FGPIOE)
#define FGPIOE_PDDR           FGPIO_PDDR_REG(FGPIOE)

/*!
 * @}
 */

/* /* end of group GPIO_Register_Accessor_Macros */

/*!
 * @}
 */

/* /* end of group GPIO_Peripheral_Access_Layer */

/*
-----
```

```

-- FTFA Peripheral Access Layer
----- */
/*!
* @addtogroup FTFA_Peripheral_Access_Layer FTFA Peripheral Access Layer
* @{
*/
/** FTFA - Register Layout Typedef */
typedef struct {
    __IO uint8_t FSTAT;           /**< Flash Status Register, offset: 0x0 */
    __IO uint8_t FCNFG;          /**< Flash Configuration Register, offset: 0x1 */
    __I  uint8_t FSEC;           /**< Flash Security Register, offset: 0x2 */
    __I  uint8_t FOPT;           /**< Flash Option Register, offset: 0x3 */
    __IO uint8_t FCCOB3;         /**< Flash Common Command Object Registers, offset: 0x4 */
    __IO uint8_t FCCOB2;         /**< Flash Common Command Object Registers, offset: 0x5 */
    __IO uint8_t FCCOB1;         /**< Flash Common Command Object Registers, offset: 0x6 */
    __IO uint8_t FCCOB0;         /**< Flash Common Command Object Registers, offset: 0x7 */
    __IO uint8_t FCCOB7;         /**< Flash Common Command Object Registers, offset: 0x8 */
    __IO uint8_t FCCOB6;         /**< Flash Common Command Object Registers, offset: 0x9 */
    __IO uint8_t FCCOB5;         /**< Flash Common Command Object Registers, offset: 0xA */
    __IO uint8_t FCCOB4;         /**< Flash Common Command Object Registers, offset: 0xB */
    __IO uint8_t FCCOB3B;        /**< Flash Common Command Object Registers, offset: 0xC */
    __IO uint8_t FCCOB2A;        /**< Flash Common Command Object Registers, offset: 0xD */
    __IO uint8_t FCCOB9;         /**< Flash Common Command Object Registers, offset: 0xE */
    __IO uint8_t FCCOB8;         /**< Flash Common Command Object Registers, offset: 0xF */
    __IO uint8_t FPROT3;          /**< Program Flash Protection Registers, offset: 0x10 */
    __IO uint8_t FPROT2;          /**< Program Flash Protection Registers, offset: 0x11 */
    __IO uint8_t FPROT1;          /**< Program Flash Protection Registers, offset: 0x12 */
}

```

```
    __IO uint8_t FPROTO;           /**< Program Flash Protection Registers, offset: 0x13 */  
} FTFA_Type, *FTFA_MemMapPtr;
```

```
/* -----  
-- FTFA - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros  
* @{  
*/
```

```
/* FTFA - Register accessors */  
  
#define FTFA_FSTAT_REG(base)          ((base)->FSTAT)  
#define FTFA_FCNG_REG(base)           ((base)->FCNFG)  
#define FTFA_FSEC_REG(base)           ((base)->FSEC)  
#define FTFA_FOPT_REG(base)           ((base)->FOPT)  
#define FTFA_FCCOB3_REG(base)         ((base)->FCCOB3)  
#define FTFA_FCCOB2_REG(base)         ((base)->FCCOB2)  
#define FTFA_FCCOB1_REG(base)         ((base)->FCCOB1)  
#define FTFA_FCCOB0_REG(base)         ((base)->FCCOB0)  
#define FTFA_FCCOB7_REG(base)         ((base)->FCCOB7)  
#define FTFA_FCCOB6_REG(base)         ((base)->FCCOB6)  
#define FTFA_FCCOB5_REG(base)         ((base)->FCCOB5)  
#define FTFA_FCCOB4_REG(base)         ((base)->FCCOB4)  
#define FTFA_FCCOB_B_REG(base)        ((base)->FCCOB_B)  
#define FTFA_FCCOB_A_REG(base)        ((base)->FCCOB_A)  
#define FTFA_FCCOB9_REG(base)         ((base)->FCCOB9)
```

```
#define FTFA_FCCOB8_REG(base) ((base)->FCCOB8)
#define FTFA_FPROT3_REG(base) ((base)->FPROT3)
#define FTFA_FPROT2_REG(base) ((base)->FPROT2)
#define FTFA_FPROT1_REG(base) ((base)->FPROT1)
#define FTFA_FPROT0_REG(base) ((base)->FPROT0)
```

```
/*
 * @}
 */
/* /* end of group FTFA_Register_Accessor_Macros */
```

```
/*
-----  
-- FTFA Register Masks  
----- */
```

```
/*
 * @addtogroup FTFA_Register_Masks FTFA Register Masks
 * @{
 */

```

```
/* FSTAT Bit Fields */

#define FTFA_FSTAT_MGSTAT0_MASK 0x1u
#define FTFA_FSTAT_MGSTAT0_SHIFT 0
#define FTFA_FSTAT_MGSTAT0_WIDTH 1

#define FTFA_FSTAT_MGSTAT0(x) (((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_MGSTAT0_SHIFT))&FTFA_FSTAT_MGSTAT0_MASK)

#define FTFA_FSTAT_FPVOL_MASK 0x10u
#define FTFA_FSTAT_FPVOL_SHIFT 4
#define FTFA_FSTAT_FPVOL_WIDTH 1
```

```

#define FTFA_FSTAT_FPVIOL(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_FPVIOL_SHIFT))&FTFA_FSTAT_FPVIOL_MASK)

#define FTFA_FSTAT_ACCERR_MASK          0x20u
#define FTFA_FSTAT_ACCERR_SHIFT        5
#define FTFA_FSTAT_ACCERR_WIDTH        1

#define FTFA_FSTAT_ACCERR(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_ACCERR_SHIFT))&FTFA_FSTAT_ACCERR_MASK)

#define FTFA_FSTAT_RDCOLERR_MASK       0x40u
#define FTFA_FSTAT_RDCOLERR_SHIFT     6
#define FTFA_FSTAT_RDCOLERR_WIDTH     1

#define FTFA_FSTAT_RDCOLERR(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_RDCOLERR_SHIFT))&FTFA_FSTAT_RDCOLERR_MASK)

#define FTFA_FSTAT_CCIF_MASK           0x80u
#define FTFA_FSTAT_CCIF_SHIFT         7
#define FTFA_FSTAT_CCIF_WIDTH         1

#define FTFA_FSTAT_CCIF(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_CCIF_SHIFT))&FTFA_FSTAT_CCIF_MASK)

/* FCNFG Bit Fields */

#define FTFA_FCNFG_ERSSUSP_MASK        0x10u
#define FTFA_FCNFG_ERSSUSP_SHIFT      4
#define FTFA_FCNFG_ERSSUSP_WIDTH      1

#define FTFA_FCNFG_ERSSUSP(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCNFG_ERSSUSP_SHIFT))&FTFA_FCNFG_ERSSUSP_MASK)

#define FTFA_FCNFG_ERSAREQ_MASK        0x20u
#define FTFA_FCNFG_ERSAREQ_SHIFT      5
#define FTFA_FCNFG_ERSAREQ_WIDTH      1

#define FTFA_FCNFG_ERSAREQ(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCNFG_ERSAREQ_SHIFT))&FTFA_FCNFG_ERSAREQ_MASK)

#define FTFA_FCNFG_RDCOLLIE_MASK       0x40u
#define FTFA_FCNFG_RDCOLLIE_SHIFT     6
#define FTFA_FCNFG_RDCOLLIE_WIDTH     1

```

```

#define FTFA_FCNFG_RDCOLLIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_RDCOLLIE_SHIFT))&FTFA_FCNFG_RDCOLLIE_MASK)

#define FTFA_FCNFG_CCIE_MASK          0x80u
#define FTFA_FCNFG_CCIE_SHIFT        7
#define FTFA_FCNFG_CCIE_WIDTH        1

#define FTFA_FCNFG_CCIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_CCIE_SHIFT))&FTFA_FCNFG_CCIE_MASK)

/* FSEC Bit Fields */

#define FTFA_FSEC_SEC_MASK           0x3u
#define FTFA_FSEC_SEC_SHIFT          0
#define FTFA_FSEC_SEC_WIDTH          2

#define FTFA_FSEC_SEC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_SEC_SHIFT))&FTFA_FSEC_SEC_MASK)

#define FTFA_FSEC_FSLACC_MASK         0xCu
#define FTFA_FSEC_FSLACC_SHIFT       2
#define FTFA_FSEC_FSLACC_WIDTH       2

#define FTFA_FSEC_FSLACC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_FSLACC_SHIFT))&FTFA_FSEC_FSLACC_MASK)

#define FTFA_FSEC_MEEN_MASK          0x30u
#define FTFA_FSEC_MEEN_SHIFT          4
#define FTFA_FSEC_MEEN_WIDTH          2

#define FTFA_FSEC_MEEN(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_MEEN_SHIFT))&FTFA_FSEC_MEEN_MASK)

#define FTFA_FSEC_KEYEN_MASK          0xC0u
#define FTFA_FSEC_KEYEN_SHIFT          6
#define FTFA_FSEC_KEYEN_WIDTH          2

#define FTFA_FSEC_KEYEN(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_KEYEN_SHIFT))&FTFA_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define FTFA_FOPT_OPT_MASK           0xFFu
#define FTFA_FOPT_OPT_SHIFT          0

```

```

#define FTFA_FOPT_OPT_WIDTH          8
#define FTFA_FOPT_OPT(x) (((uint8_t)((uint8_t)(x)<<FTFA_FOPT_OPT_SHIFT))&FTFA_FOPT_OPT_MASK)

/* FCCOB3 Bit Fields */

#define FTFA_FCCOB3_CCOBn_MASK        0xFFu
#define FTFA_FCCOB3_CCOBn_SHIFT       0
#define FTFA_FCCOB3_CCOBn_WIDTH       8
#define FTFA_FCCOB3_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB3_CCOBn_SHIFT))&FTFA_FCCOB3_CCOBn_MASK)

/* FCCOB2 Bit Fields */

#define FTFA_FCCOB2_CCOBn_MASK        0xFFu
#define FTFA_FCCOB2_CCOBn_SHIFT       0
#define FTFA_FCCOB2_CCOBn_WIDTH       8
#define FTFA_FCCOB2_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB2_CCOBn_SHIFT))&FTFA_FCCOB2_CCOBn_MASK)

/* FCCOB1 Bit Fields */

#define FTFA_FCCOB1_CCOBn_MASK        0xFFu
#define FTFA_FCCOB1_CCOBn_SHIFT       0
#define FTFA_FCCOB1_CCOBn_WIDTH       8
#define FTFA_FCCOB1_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB1_CCOBn_SHIFT))&FTFA_FCCOB1_CCOBn_MASK)

/* FCCOB0 Bit Fields */

#define FTFA_FCCOB0_CCOBn_MASK        0xFFu
#define FTFA_FCCOB0_CCOBn_SHIFT       0
#define FTFA_FCCOB0_CCOBn_WIDTH       8
#define FTFA_FCCOB0_CCOBn(x) (((uint8_t)((uint8_t)(x)<<FTFA_FCCOB0_CCOBn_SHIFT))&FTFA_FCCOB0_CCOBn_MASK)

/* FCCOB7 Bit Fields */

#define FTFA_FCCOB7_CCOBn_MASK        0xFFu
#define FTFA_FCCOB7_CCOBn_SHIFT       0

```

```

#define FTFA_FCCOB7_CCOBn_WIDTH      8
#define FTFA_FCCOB7_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB7_CCOBn_SHIFT))&FTFA_FCCOB7_CCOBn_MASK)

/* FCCOB6 Bit Fields */

#define FTFA_FCCOB6_CCOBn_MASK      0xFFu
#define FTFA_FCCOB6_CCOBn_SHIFT     0
#define FTFA_FCCOB6_CCOBn_WIDTH     8
#define FTFA_FCCOB6_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB6_CCOBn_SHIFT))&FTFA_FCCOB6_CCOBn_MASK)

/* FCCOB5 Bit Fields */

#define FTFA_FCCOB5_CCOBn_MASK      0xFFu
#define FTFA_FCCOB5_CCOBn_SHIFT     0
#define FTFA_FCCOB5_CCOBn_WIDTH     8
#define FTFA_FCCOB5_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB5_CCOBn_SHIFT))&FTFA_FCCOB5_CCOBn_MASK)

/* FCCOB4 Bit Fields */

#define FTFA_FCCOB4_CCOBn_MASK      0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT     0
#define FTFA_FCCOB4_CCOBn_WIDTH     8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK)

/* FCCOBB Bit Fields */

#define FTFA_FCCOBB_CCOBn_MASK      0xFFu
#define FTFA_FCCOBB_CCOBn_SHIFT     0
#define FTFA_FCCOBB_CCOBn_WIDTH     8
#define FTFA_FCCOBB_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOBB_CCOBn_SHIFT))&FTFA_FCCOBB_CCOBn_MASK)

/* FCCOBA Bit Fields */

#define FTFA_FCCOBA_CCOBn_MASK      0xFFu
#define FTFA_FCCOBA_CCOBn_SHIFT     0

```

```

#define FTFA_FCCOBA_CCOBn_WIDTH          8
#define FTFA_FCCOBA_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOBA_CCOBn_SHIFT))&FTFA_FCCOBA_CCOBn_MASK)

/* FCCOB9 Bit Fields */

#define FTFA_FCCOB9_CCOBn_MASK           0xFFu
#define FTFA_FCCOB9_CCOBn_SHIFT          0
#define FTFA_FCCOB9_CCOBn_WIDTH          8
#define FTFA_FCCOB9_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB9_CCOBn_SHIFT))&FTFA_FCCOB9_CCOBn_MASK)

/* FCCOB8 Bit Fields */

#define FTFA_FCCOB8_CCOBn_MASK           0xFFu
#define FTFA_FCCOB8_CCOBn_SHIFT          0
#define FTFA_FCCOB8_CCOBn_WIDTH          8
#define FTFA_FCCOB8_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB8_CCOBn_SHIFT))&FTFA_FCCOB8_CCOBn_MASK)

/* FPROT3 Bit Fields */

#define FTFA_FPROT3_PROT_MASK            0xFFu
#define FTFA_FPROT3_PROT_SHIFT           0
#define FTFA_FPROT3_PROT_WIDTH          8
#define FTFA_FPROT3_PROT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

#define FTFA_FPROT2_PROT_MASK            0xFFu
#define FTFA_FPROT2_PROT_SHIFT           0
#define FTFA_FPROT2_PROT_WIDTH          8
#define FTFA_FPROT2_PROT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FPROT2_PROT_SHIFT))&FTFA_FPROT2_PROT_MASK)

/* FPROT1 Bit Fields */

#define FTFA_FPROT1_PROT_MASK            0xFFu
#define FTFA_FPROT1_PROT_SHIFT           0

```

```

#define FTFA_FPROT1_PROT_WIDTH          8
#define FTFA_FPROT1_PROT(x)             (((uint8_t)((uint8_t)(x)<<FTFA_FPROT1_PROT_SHIFT))&FTFA_FPROT1_PROT_MASK)
/* FPROT0 Bit Fields */
#define FTFA_FPROT0_PROT_MASK          0xFFu
#define FTFA_FPROT0_PROT_SHIFT         0
#define FTFA_FPROT0_PROT_WIDTH         8
#define FTFA_FPROT0_PROT(x)            (((uint8_t)((uint8_t)(x)<<FTFA_FPROT0_PROT_SHIFT))&FTFA_FPROT0_PROT_MASK)

/*!
 * @}
 */

/* /* end of group FTFA_Register_Masks */

```

```

/* FTFA - Peripheral instance base addresses */
/** Peripheral FTFA base address */
#define FTFA_BASE                  (0x40020000u)
/** Peripheral FTFA base pointer */
#define FTFA           ((FTFA_Type *)FTFA_BASE)
#define FTFA_BASE_PTR      (FTFA)
/** Array initializer of FTFA peripheral base addresses */
#define FTFA_BASE_ADDRS     { FTFA_BASE }
/** Array initializer of FTFA peripheral base pointers */
#define FTFA_BASE_PTRS      { FTFA }

```

```

/*
-----*
-- FTFA - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros  
 * @{  
 */  
  
/* FTFA - Register instance definitions */  
/* FTFA */  
  
#define FTFA_FSTAT           FTFA_FSTAT_REG(FTFA)  
#define FTFA_FCNG            FTFA_FCNG_REG(FTFA)  
#define FTFA_FSEC            FTFA_FSEC_REG(FTFA)  
#define FTFA_FOPT            FTFA_FOPT_REG(FTFA)  
#define FTFA_FCCOB3          FTFA_FCCOB3_REG(FTFA)  
#define FTFA_FCCOB2          FTFA_FCCOB2_REG(FTFA)  
#define FTFA_FCCOB1          FTFA_FCCOB1_REG(FTFA)  
#define FTFA_FCCOB0          FTFA_FCCOB0_REG(FTFA)  
#define FTFA_FCCOB7          FTFA_FCCOB7_REG(FTFA)  
#define FTFA_FCCOB6          FTFA_FCCOB6_REG(FTFA)  
#define FTFA_FCCOB5          FTFA_FCCOB5_REG(FTFA)  
#define FTFA_FCCOB4          FTFA_FCCOB4_REG(FTFA)  
#define FTFA_FCCOBB          FTFA_FCCOBB_REG(FTFA)  
#define FTFA_FCCOBA          FTFA_FCCOBA_REG(FTFA)  
#define FTFA_FCCOB9          FTFA_FCCOB9_REG(FTFA)  
#define FTFA_FCCOB8          FTFA_FCCOB8_REG(FTFA)  
#define FTFA_FPROT3          FTFA_FPROT3_REG(FTFA)  
#define FTFA_FPROT2          FTFA_FPROT2_REG(FTFA)  
#define FTFA_FPROT1          FTFA_FPROT1_REG(FTFA)  
#define FTFA_FPROTO          FTFA_FPROTO_REG(FTFA)
```

```
/*!  
 * @}  
 */ /* end of group FTFA_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group FTFA_Peripheral_Access_Layer */
```

```
/* -----  
-- GPIO Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access Layer  
 * @{  
 */
```

```
/** GPIO - Register Layout Typedef */  
  
typedef struct {  
  
    __IO uint32_t PDOR;           /**< Port Data Output Register, offset: 0x0 */  
    __O uint32_t PSOR;           /**< Port Set Output Register, offset: 0x4 */  
    __O uint32_t PCOR;           /**< Port Clear Output Register, offset: 0x8 */  
    __O uint32_t PTOR;           /**< Port Toggle Output Register, offset: 0xC */  
    __I uint32_t PDIR;           /**< Port Data Input Register, offset: 0x10 */  
    __IO uint32_t PDDR;           /**< Port Data Direction Register, offset: 0x14 */  
  
} GPIO_Type, *GPIO_MemMapPtr;
```

```
/* -----
-- GPIO - Register accessor macros
----- */

/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{
*
/* GPIO - Register accessors */
#define GPIO_PDOR_REG(base)      ((base)->PDOR)
#define GPIO_PSOR_REG(base)      ((base)->PSOR)
#define GPIO_PCOR_REG(base)      ((base)->PCOR)
#define GPIO_PTOR_REG(base)      ((base)->PTOR)
#define GPIO_PDIR_REG(base)      ((base)->PDIR)
#define GPIO_PDDR_REG(base)      ((base)->PDDR)

*/
* @}
*/
/* end of group GPIO_Register_Accessor_Macros */
```

```
/* -----
-- GPIO Register Masks
----- */

/*!
```

```

* @addtogroup GPIO_Register_Masks GPIO Register Masks

* @{

/*
 * PDOR Bit Fields
 */
#define GPIO_PDOR_PDO_MASK          0xFFFFFFFFu
#define GPIO_PDOR_PDO_SHIFT         0
#define GPIO_PDOR_PDO_WIDTH         32
#define GPIO_PDOR_PDO(x)            (((uint32_t)((uint32_t)(x)<<GPIO_PDOR_PDO_SHIFT))&GPIO_PDOR_PDO_MASK)

/*
 * PSOR Bit Fields
 */
#define GPIO_PSOR PTSO MASK        0xFFFFFFFFu
#define GPIO_PSOR PTSO SHIFT       0
#define GPIO_PSOR PTSO WIDTH       32
#define GPIO_PSOR PTSO(x)          (((uint32_t)((uint32_t)(x)<<GPIO_PSOR_PTSO_SHIFT))&GPIO_PSOR_PTSO_MASK)

/*
 * PCOR Bit Fields
 */
#define GPIO_PCOR PTCO MASK        0xFFFFFFFFu
#define GPIO_PCOR PTCO SHIFT       0
#define GPIO_PCOR PTCO WIDTH       32
#define GPIO_PCOR PTCO(x)          (((uint32_t)((uint32_t)(x)<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)

/*
 * PTOR Bit Fields
 */
#define GPIO_PTOR PTTO MASK        0xFFFFFFFFu
#define GPIO_PTOR PTTO SHIFT       0
#define GPIO_PTOR PTTO WIDTH       32
#define GPIO_PTOR PTTO(x)          (((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/*
 * PDIR Bit Fields
 */
#define GPIO_PDIR PDI MASK         0xFFFFFFFFu

```

```

#define GPIO_PDIR_PDI_SHIFT          0
#define GPIO_PDIR_PDI_WIDTH          32
#define GPIO_PDIR_PDI(x)             (((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK           0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT          0
#define GPIO_PDDR_PDD_WIDTH          32
#define GPIO_PDDR_PDD(x)             (((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

/*!
 * @}
 */

/* /* end of group GPIO_Register_Masks */

```

```

/* GPIO - Peripheral instance base addresses */

/** Peripheral GPIOA base address */

#define GPIOA_BASE                  (0x400FF000u)

/** Peripheral GPIOA base pointer */

#define GPIOA                       ((GPIO_Type *)GPIOA_BASE)

#define GPIOA_BASE_PTR              (GPIOA)

/** Peripheral GPIOB base address */

#define GPIOB_BASE                  (0x400FF040u)

/** Peripheral GPIOB base pointer */

#define GPIOB                       ((GPIO_Type *)GPIOB_BASE)

#define GPIOB_BASE_PTR              (GPIOB)

/** Peripheral GPIOC base address */

#define GPIOC_BASE                  (0x400FF080u)

```

```

/** Peripheral GPIOC base pointer */

#define GPIOC           ((GPIO_Type *)GPIOC_BASE)

#define GPIOC_BASE_PTR      (GPIOC)

/** Peripheral GPIOD base address */

#define GPIOD_BASE        (0x400FF0C0u)

/** Peripheral GPIOD base pointer */

#define GPIOD           ((GPIO_Type *)GPIOD_BASE)

#define GPIOD_BASE_PTR      (GPIOD)

/** Peripheral GPIOE base address */

#define GPIOE_BASE        (0x400FF100u)

/** Peripheral GPIOE base pointer */

#define GPIOE           ((GPIO_Type *)GPIOE_BASE)

#define GPIOE_BASE_PTR      (GPIOE)

/** Array initializer of GPIO peripheral base addresses */

#define GPIO_BASE_ADDRS      { GPIOA_BASE, GPIOB_BASE, GPIOC_BASE, GPIOD_BASE,
                           GPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */

#define GPIO_BASE_PTRS       { GPIOA, GPIOB, GPIOC, GPIOD, GPIOE }

/*
-----*
-- GPIO - Register accessor macros
----- */
/*!
 * @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
 * @{
 */

```

```
/* GPIO - Register instance definitions */

/* GPIOA */

#define GPIOA_PDOR          GPIO_PDOR_REG(GPIOA)
#define GPIOA_PSOR          GPIO_PSOR_REG(GPIOA)
#define GPIOA_PCOR          GPIO_PCOR_REG(GPIOA)
#define GPIOA_PTOR          GPIO_PTOR_REG(GPIOA)
#define GPIOA_PDIR          GPIO_PDIR_REG(GPIOA)
#define GPIOA_PDDR          GPIO_PDDR_REG(GPIOA)

/* GPIOB */

#define GPIOB_PDOR          GPIO_PDOR_REG(GPIOB)
#define GPIOB_PSOR          GPIO_PSOR_REG(GPIOB)
#define GPIOB_PCOR          GPIO_PCOR_REG(GPIOB)
#define GPIOB_PTOR          GPIO_PTOR_REG(GPIOB)
#define GPIOB_PDIR          GPIO_PDIR_REG(GPIOB)
#define GPIOB_PDDR          GPIO_PDDR_REG(GPIOB)

/* GPIOC */

#define GPIOC_PDOR          GPIO_PDOR_REG(GPIOC)
#define GPIOC_PSOR          GPIO_PSOR_REG(GPIOC)
#define GPIOC_PCOR          GPIO_PCOR_REG(GPIOC)
#define GPIOC_PTOR          GPIO_PTOR_REG(GPIOC)
#define GPIOC_PDIR          GPIO_PDIR_REG(GPIOC)
#define GPIOC_PDDR          GPIO_PDDR_REG(GPIOC)

/* GPIOD */

#define GPIOD_PDOR          GPIO_PDOR_REG(GPIOD)
#define GPIOD_PSOR          GPIO_PSOR_REG(GPIOD)
#define GPIOD_PCOR          GPIO_PCOR_REG(GPIOD)
#define GPIOD_PTOR          GPIO_PTOR_REG(GPIOD)
#define GPIOD_PDIR          GPIO_PDIR_REG(GPIOD)
#define GPIOD_PDDR          GPIO_PDDR_REG(GPIOD)
```

```
/* GPIOE */

#define GPIOE_PDOR          GPIO_PDOR_REG(GPIOE)
#define GPIOE_PSOR          GPIO_PSOR_REG(GPIOE)
#define GPIOE_PCOR          GPIO_PCOR_REG(GPIOE)
#define GPIOE_PTOR          GPIO_PTOR_REG(GPIOE)
#define GPIOE_PDIR          GPIO_PDIR_REG(GPIOE)
#define GPIOE_PDDR          GPIO_PDDR_REG(GPIOE)
```

```
/*
 * @}
 */
/* end of group GPIO_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group GPIO_Peripheral_Access_Layer */
```

```
/* -----
-- I2C Peripheral Access Layer
----- */
```

```
/*
 * @addtogroup I2C_Peripheral_Access_Layer I2C Peripheral Access Layer
 * @{
 */

```

```
/** I2C - Register Layout Typedef */
```

```
typedef struct {
```

```

__IO uint8_t A1;           /**< I2C Address Register 1, offset: 0x0 */
__IO uint8_t F;            /**< I2C Frequency Divider register, offset: 0x1 */
__IO uint8_t C1;           /**< I2C Control Register 1, offset: 0x2 */
__IO uint8_t S;            /**< I2C Status register, offset: 0x3 */
__IO uint8_t D;            /**< I2C Data I/O register, offset: 0x4 */
__IO uint8_t C2;           /**< I2C Control Register 2, offset: 0x5 */
__IO uint8_t FLT;          /**< I2C Programmable Input Glitch Filter register, offset: 0x6 */
__IO uint8_t RA;           /**< I2C Range Address register, offset: 0x7 */
__IO uint8_t SMB;          /**< I2C SMBus Control and Status register, offset: 0x8 */
__IO uint8_t A2;           /**< I2C Address Register 2, offset: 0x9 */
__IO uint8_t SLTH;          /**< I2C SCL Low Timeout Register High, offset: 0xA */
__IO uint8_t SLTL;          /**< I2C SCL Low Timeout Register Low, offset: 0xB */

} I2C_Type, *I2C_MemMapPtr;

```

```

/*
-----*
-- I2C - Register accessor macros
----- */

```

```

/*!
* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros
* @{
*/

```

```

/* I2C - Register accessors */

#define I2C_A1_REG(base)          ((base)->A1)
#define I2C_F_REG(base)           ((base)->F)
#define I2C_C1_REG(base)          ((base)->C1)
#define I2C_S_REG(base)           ((base)->S)

```

```

#define I2C_D_REG(base)          ((base)->D)
#define I2C_C2_REG(base)         ((base)->C2)
#define I2C_FLT_REG(base)        ((base)->FLT)
#define I2C_RA_REG(base)         ((base)->RA)
#define I2C_SMB_REG(base)        ((base)->SMB)
#define I2C_A2_REG(base)         ((base)->A2)
#define I2C_SLTH_REG(base)       ((base)->SLTH)
#define I2C_SLTL_REG(base)       ((base)->SLTL)

/*!
 * @}
 */

/* /* end of group I2C_Register_Accessor_Macros */

```

```

/*
-----*
-- I2C Register Masks
----- */

```

```

/*!

* @addtogroup I2C_Register_Masks I2C Register Masks
* @{
*/

```

```

/* A1 Bit Fields */

#define I2C_A1_AD_MASK           0xFEu
#define I2C_A1_AD_SHIFT          1
#define I2C_A1_AD_WIDTH          7

#define I2C_A1_AD(x)
(((uint8_t)((uint8_t)(x)<<I2C_A1_AD_SHIFT))&I2C_A1_AD_MASK)

```

```

/* F Bit Fields */

#define I2C_F_ICR_MASK          0x3Fu
#define I2C_F_ICR_SHIFT         0
#define I2C_F_ICR_WIDTH         6
#define I2C_F_ICR(x)            (((uint8_t)((uint8_t)(x)<<I2C_F_ICR_SHIFT))&I2C_F_ICR_MASK)

#define I2C_F_MULT_MASK         0xC0u
#define I2C_F_MULT_SHIFT        6
#define I2C_F_MULT_WIDTH        2
#define I2C_F_MULT(x)           (((uint8_t)((uint8_t)(x)<<I2C_F_MULT_SHIFT))&I2C_F_MULT_MASK)

/* C1 Bit Fields */

#define I2C_C1_DMAEN_MASK       0x1u
#define I2C_C1_DMAEN_SHIFT      0
#define I2C_C1_DMAEN_WIDTH      1
#define I2C_C1_DMAEN(x)         (((uint8_t)((uint8_t)(x)<<I2C_C1_DMAEN_SHIFT))&I2C_C1_DMAEN_MASK)

#define I2C_C1_WUEN_MASK        0x2u
#define I2C_C1_WUEN_SHIFT       1
#define I2C_C1_WUEN_WIDTH       1
#define I2C_C1_WUEN(x)          (((uint8_t)((uint8_t)(x)<<I2C_C1_WUEN_SHIFT))&I2C_C1_WUEN_MASK)

#define I2C_C1_RSTA_MASK        0x4u
#define I2C_C1_RSTA_SHIFT       2
#define I2C_C1_RSTA_WIDTH       1
#define I2C_C1_RSTA(x)          (((uint8_t)((uint8_t)(x)<<I2C_C1_RSTA_SHIFT))&I2C_C1_RSTA_MASK)

#define I2C_C1_TXAK_MASK        0x8u
#define I2C_C1_TXAK_SHIFT       3
#define I2C_C1_TXAK_WIDTH       1

```

```

#define I2C_C1_TXAK(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_TXAK_SHIFT))&I2C_C1_TXAK_MASK)

#define I2C_C1_TX_MASK          0x10u
#define I2C_C1_TX_SHIFT         4
#define I2C_C1_TX_WIDTH         1
#define I2C_C1_TX(x)            (((uint8_t)((uint8_t)(x)<<I2C_C1_TX_SHIFT))&I2C_C1_TX_MASK)

#define I2C_C1_MST_MASK         0x20u
#define I2C_C1_MST_SHIFT        5
#define I2C_C1_MST_WIDTH        1
#define I2C_C1_MST(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_MST_SHIFT))&I2C_C1_MST_MASK)

#define I2C_C1_IICIE_MASK       0x40u
#define I2C_C1_IICIE_SHIFT      6
#define I2C_C1_IICIE_WIDTH      1
#define I2C_C1_IICIE(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_IICIE_SHIFT))&I2C_C1_IICIE_MASK)

#define I2C_C1_IICEN_MASK       0x80u
#define I2C_C1_IICEN_SHIFT      7
#define I2C_C1_IICEN_WIDTH      1
#define I2C_C1_IICEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_C1_IICEN_SHIFT))&I2C_C1_IICEN_MASK)

/* S Bit Fields */

#define I2C_S_RXAK_MASK          0x1u
#define I2C_S_RXAK_SHIFT         0
#define I2C_S_RXAK_WIDTH         1
#define I2C_S_RXAK(x)
(((uint8_t)((uint8_t)(x)<<I2C_S_RXAK_SHIFT))&I2C_S_RXAK_MASK)

#define I2C_S_IICIF_MASK         0x2u
#define I2C_S_IICIF_SHIFT        1
#define I2C_S_IICIF_WIDTH        1

```

```

#define I2C_S_IICIF(x)          (((uint8_t)((uint8_t)(x)<<I2C_S_IICIF_SHIFT))&I2C_S_IICIF_MASK)
#define I2C_S_SRW_MASK          0x4u
#define I2C_S_SRW_SHIFT         2
#define I2C_S_SRW_WIDTH         1
#define I2C_S_SRW(x)            (((uint8_t)((uint8_t)(x)<<I2C_S_SRW_SHIFT))&I2C_S_SRW_MASK)

#define I2C_S_RAM_MASK          0x8u
#define I2C_S_RAM_SHIFT         3
#define I2C_S_RAM_WIDTH         1
#define I2C_S_RAM(x)             (((uint8_t)((uint8_t)(x)<<I2C_S_RAM_SHIFT))&I2C_S_RAM_MASK)

#define I2C_S_ARBL_MASK         0x10u
#define I2C_S_ARBL_SHIFT        4
#define I2C_S_ARBL_WIDTH        1
#define I2C_S_ARBL(x)            (((uint8_t)((uint8_t)(x)<<I2C_S_ARBL_SHIFT))&I2C_S_ARBL_MASK)

#define I2C_S_BUSY_MASK          0x20u
#define I2C_S_BUSY_SHIFT         5
#define I2C_S_BUSY_WIDTH         1
#define I2C_S_BUSY(x)             (((uint8_t)((uint8_t)(x)<<I2C_S_BUSY_SHIFT))&I2C_S_BUSY_MASK)

#define I2C_S_IAAS_MASK          0x40u
#define I2C_S_IAAS_SHIFT         6
#define I2C_S_IAAS_WIDTH         1
#define I2C_S_IAAS(x)             (((uint8_t)((uint8_t)(x)<<I2C_S_IAAS_SHIFT))&I2C_S_IAAS_MASK)

#define I2C_S_TCF_MASK           0x80u
#define I2C_S_TCF_SHIFT          7
#define I2C_S_TCF_WIDTH          1
#define I2C_S_TCF(x)              (((uint8_t)((uint8_t)(x)<<I2C_S_TCF_SHIFT))&I2C_S_TCF_MASK)

/* D Bit Fields */

```

```

#define I2C_D_DATA_MASK          0xFFu
#define I2C_D_DATA_SHIFT         0
#define I2C_D_DATA_WIDTH         8
#define I2C_D_DATA(x)            (((uint8_t)((uint8_t)(x))<<I2C_D_DATA_SHIFT))&I2C_D_DATA_MASK

/* C2 Bit Fields */

#define I2C_C2_AD_MASK           0x7u
#define I2C_C2_AD_SHIFT          0
#define I2C_C2_AD_WIDTH          3
#define I2C_C2_AD(x)              (((uint8_t)((uint8_t)(x))<<I2C_C2_AD_SHIFT))&I2C_C2_AD_MASK

#define I2C_C2_RMEN_MASK          0x8u
#define I2C_C2_RMEN_SHIFT         3
#define I2C_C2_RMEN_WIDTH         1
#define I2C_C2_RMEN(x)             (((uint8_t)((uint8_t)(x))<<I2C_C2_RMEN_SHIFT))&I2C_C2_RMEN_MASK

#define I2C_C2_SBRC_MASK          0x10u
#define I2C_C2_SBRC_SHIFT         4
#define I2C_C2_SBRC_WIDTH          1
#define I2C_C2_SBRC(x)              (((uint8_t)((uint8_t)(x))<<I2C_C2_SBRC_SHIFT))&I2C_C2_SBRC_MASK

#define I2C_C2_HDRS_MASK           0x20u
#define I2C_C2_HDRS_SHIFT          5
#define I2C_C2_HDRS_WIDTH          1
#define I2C_C2_HDRS(x)              (((uint8_t)((uint8_t)(x))<<I2C_C2_HDRS_SHIFT))&I2C_C2_HDRS_MASK

#define I2C_C2_ADEXT_MASK          0x40u
#define I2C_C2_ADEXT_SHIFT         6
#define I2C_C2_ADEXT_WIDTH          1
#define I2C_C2_ADEXT(x)              (((uint8_t)((uint8_t)(x))<<I2C_C2_ADEXT_SHIFT))&I2C_C2_ADEXT_MASK

```

```

#define I2C_C2_GCAEN_MASK          0x80u
#define I2C_C2_GCAEN_SHIFT         7
#define I2C_C2_GCAEN_WIDTH         1
#define I2C_C2_GCAEN(x) (((uint8_t)((uint8_t)(x))<<I2C_C2_GCAEN_SHIFT))&I2C_C2_GCAEN_MASK)

/* FLT Bit Fields */

#define I2C_FLT_FLT_MASK           0x1Fu
#define I2C_FLT_FLT_SHIFT          0
#define I2C_FLT_FLT_WIDTH          5
#define I2C_FLT_FLT(x) (((uint8_t)((uint8_t)(x))<<I2C_FLT_FLT_SHIFT))&I2C_FLT_FLT_MASK)

#define I2C_FLT_STOPIE_MASK         0x20u
#define I2C_FLT_STOPIE_SHIFT        5
#define I2C_FLT_STOPIE_WIDTH        1
#define I2C_FLT_STOPIE(x) (((uint8_t)((uint8_t)(x))<<I2C_FLT_STOPIE_SHIFT))&I2C_FLT_STOPIE_MASK)

#define I2C_FLT_STOPF_MASK          0x40u
#define I2C_FLT_STOPF_SHIFT         6
#define I2C_FLT_STOPF_WIDTH         1
#define I2C_FLT_STOPF(x) (((uint8_t)((uint8_t)(x))<<I2C_FLT_STOPF_SHIFT))&I2C_FLT_STOPF_MASK)

#define I2C_FLT_SHEN_MASK           0x80u
#define I2C_FLT_SHEN_SHIFT          7
#define I2C_FLT_SHEN_WIDTH          1
#define I2C_FLT_SHEN(x) (((uint8_t)((uint8_t)(x))<<I2C_FLT_SHEN_SHIFT))&I2C_FLT_SHEN_MASK)

/* RA Bit Fields */

#define I2C_RA_RAD_MASK             0xFEu
#define I2C_RA_RAD_SHIFT            1
#define I2C_RA_RAD_WIDTH            7

```

```

#define I2C_RA_RAD(x)
(((uint8_t)((uint8_t)(x)<<I2C_RA_RAD_SHIFT))&I2C_RA_RAD_MASK)

/* SMB Bit Fields */

#define I2C_SMB_SHTF2IE_MASK          0x1u
#define I2C_SMB_SHTF2IE_SHIFT         0
#define I2C_SMB_SHTF2IE_WIDTH         1

#define I2C_SMB_SHTF2IE(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2IE_SHIFT))&I2C_SMB_SHTF2IE_MASK)

#define I2C_SMB_SHTF2_MASK            0x2u
#define I2C_SMB_SHTF2_SHIFT           1
#define I2C_SMB_SHTF2_WIDTH           1

#define I2C_SMB_SHTF2(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2_SHIFT))&I2C_SMB_SHTF2_MASK)

#define I2C_SMB_SHTF1_MASK            0x4u
#define I2C_SMB_SHTF1_SHIFT           2
#define I2C_SMB_SHTF1_WIDTH           1

#define I2C_SMB_SHTF1(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF1_SHIFT))&I2C_SMB_SHTF1_MASK)

#define I2C_SMB_SLTF_MASK             0x8u
#define I2C_SMB_SLTF_SHIFT            3
#define I2C_SMB_SLTF_WIDTH            1

#define I2C_SMB_SLTF(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SLTF_SHIFT))&I2C_SMB_SLTF_MASK)

#define I2C_SMB_TCKSEL_MASK           0x10u
#define I2C_SMB_TCKSEL_SHIFT          4
#define I2C_SMB_TCKSEL_WIDTH          1

#define I2C_SMB_TCKSEL(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_TCKSEL_SHIFT))&I2C_SMB_TCKSEL_MASK)

#define I2C_SMB_SIICAEN_MASK          0x20u
#define I2C_SMB_SIICAEN_SHIFT         5
#define I2C_SMB_SIICAEN_WIDTH         1

```

```

#define I2C_SMB_SIICAEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SIICAEN_SHIFT))&I2C_SMB_SIICAEN_MASK)

#define I2C_SMB_ALERTEN_MASK          0x40u
#define I2C_SMB_ALERTEN_SHIFT        6
#define I2C_SMB_ALERTEN_WIDTH        1

#define I2C_SMB_ALERTEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_ALERTEN_SHIFT))&I2C_SMB_ALERTEN_MASK)

#define I2C_SMB_FACK_MASK           0x80u
#define I2C_SMB_FACK_SHIFT          7
#define I2C_SMB_FACK_WIDTH          1

#define I2C_SMB_FACK(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_FACK_SHIFT))&I2C_SMB_FACK_MASK)

/* A2 Bit Fields */

#define I2C_A2_SAD_MASK             0xFEu
#define I2C_A2_SAD_SHIFT            1
#define I2C_A2_SAD_WIDTH            7

#define I2C_A2_SAD(x)
(((uint8_t)((uint8_t)(x)<<I2C_A2_SAD_SHIFT))&I2C_A2_SAD_MASK)

/* SLTH Bit Fields */

#define I2C_SLTH_SSLT_MASK          0xFFu
#define I2C_SLTH_SSLT_SHIFT         0
#define I2C_SLTH_SSLT_WIDTH         8

#define I2C_SLTH_SSLT(x)
(((uint8_t)((uint8_t)(x)<<I2C_SLTH_SSLT_SHIFT))&I2C_SLTH_SSLT_MASK)

/* SLTL Bit Fields */

#define I2C_SLTL_SSLT_MASK          0xFFu
#define I2C_SLTL_SSLT_SHIFT         0
#define I2C_SLTL_SSLT_WIDTH         8

#define I2C_SLTL_SSLT(x)
(((uint8_t)((uint8_t)(x)<<I2C_SLTL_SSLT_SHIFT))&I2C_SLTL_SSLT_MASK)

```

```

/*!
* @}
*/ /* end of group I2C_Register_Masks */

/* I2C - Peripheral instance base addresses */

/** Peripheral I2C0 base address */
#define I2C0_BASE          (0x40066000u)

/** Peripheral I2C0 base pointer */
#define I2C0          ((I2C_Type *)I2C0_BASE)
#define I2C0_PTR        (I2C0)

/** Peripheral I2C1 base address */
#define I2C1_BASE          (0x40067000u)

/** Peripheral I2C1 base pointer */
#define I2C1          ((I2C_Type *)I2C1_BASE)
#define I2C1_PTR        (I2C1)

/** Array initializer of I2C peripheral base addresses */
#define I2C_BASE_ADDRS      { I2C0_BASE, I2C1_BASE }

/** Array initializer of I2C peripheral base pointers */
#define I2C_BASE_PTRS       { I2C0, I2C1 }

/*
----- --
-- I2C - Register accessor macros
----- */
/*!
* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros
* @{
*/

```

```
/* I2C - Register instance definitions */

/* I2C0 */

#define I2C0_A1          I2C_A1_REG(I2C0)
#define I2C0_F           I2C_F_REG(I2C0)
#define I2C0_C1          I2C_C1_REG(I2C0)
#define I2C0_S           I2C_S_REG(I2C0)
#define I2C0_D           I2C_D_REG(I2C0)
#define I2C0_C2          I2C_C2_REG(I2C0)
#define I2C0_FLT         I2C_FLT_REG(I2C0)
#define I2C0_RA          I2C_RA_REG(I2C0)
#define I2C0_SMB         I2C_SMB_REG(I2C0)
#define I2C0_A2          I2C_A2_REG(I2C0)
#define I2C0_SLTH        I2C_SLTH_REG(I2C0)
#define I2C0_SLTL        I2C_SLTL_REG(I2C0)

/* I2C1 */

#define I2C1_A1          I2C_A1_REG(I2C1)
#define I2C1_F           I2C_F_REG(I2C1)
#define I2C1_C1          I2C_C1_REG(I2C1)
#define I2C1_S           I2C_S_REG(I2C1)
#define I2C1_D           I2C_D_REG(I2C1)
#define I2C1_C2          I2C_C2_REG(I2C1)
#define I2C1_FLT         I2C_FLT_REG(I2C1)
#define I2C1_RA          I2C_RA_REG(I2C1)
#define I2C1_SMB         I2C_SMB_REG(I2C1)
#define I2C1_A2          I2C_A2_REG(I2C1)
#define I2C1_SLTH        I2C_SLTH_REG(I2C1)
#define I2C1_SLTL        I2C_SLTL_REG(I2C1)
```

```
/*!
 * @}
 */ /* end of group I2C_Register_Accessor_Macros */
```

```
/*!
 * @}
 */ /* end of group I2C_Peripheral_Access_Layer */
```

```
/* -----
-- LLWU Peripheral Access Layer
----- */
```

```
/*!
 * @addtogroup LLWU_Peripheral_Access_Layer LLWU Peripheral Access Layer
 * @{
 */

```

```
/** LLWU - Register Layout Typedef */

typedef struct {

    __IO uint8_t PE1;           /**< LLWU Pin Enable 1 register, offset: 0x0 */
    __IO uint8_t PE2;           /**< LLWU Pin Enable 2 register, offset: 0x1 */
    __IO uint8_t PE3;           /**< LLWU Pin Enable 3 register, offset: 0x2 */
    __IO uint8_t PE4;           /**< LLWU Pin Enable 4 register, offset: 0x3 */
    __IO uint8_t ME;            /**< LLWU Module Enable register, offset: 0x4 */
    __IO uint8_t F1;            /**< LLWU Flag 1 register, offset: 0x5 */
    __IO uint8_t F2;            /**< LLWU Flag 2 register, offset: 0x6 */
}
```

```
__I uint8_t F3;          /**< LLWU Flag 3 register, offset: 0x7 */
__IO uint8_t FILT1;      /**< LLWU Pin Filter 1 register, offset: 0x8 */
__IO uint8_t FILT2;      /**< LLWU Pin Filter 2 register, offset: 0x9 */

} LLWU_Type, *LLWU_MemMapPtr;
```

```
/*
-----  
-- LLWU - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros  
* @{  
*/
```

```
/* LLWU - Register accessors */  
  
#define LLWU_PE1_REG(base)      ((base)->PE1)  
#define LLWU_PE2_REG(base)      ((base)->PE2)  
#define LLWU_PE3_REG(base)      ((base)->PE3)  
#define LLWU_PE4_REG(base)      ((base)->PE4)  
#define LLWU_ME_REG(base)       ((base)->ME)  
#define LLWU_F1_REG(base)       ((base)->F1)  
#define LLWU_F2_REG(base)       ((base)->F2)  
#define LLWU_F3_REG(base)       ((base)->F3)  
#define LLWU_FILT1_REG(base)    ((base)->FILT1)  
#define LLWU_FILT2_REG(base)    ((base)->FILT2)
```

```
/*!  
* @}
```

```
*/ /* end of group LLWU_Register_Accessor_Macros */
```

```
/* -----  
-- LLWU Register Masks  
----- */
```

```
/*!  
 * @addtogroup LLWU_Register_Masks LLWU Register Masks  
 * {@  
 */
```

```
/* PE1 Bit Fields */  
  
#define LLWU_PE1_WUPE0_MASK          0x3u  
#define LLWU_PE1_WUPE0_SHIFT         0  
#define LLWU_PE1_WUPE0_WIDTH         2  
#define LLWU_PE1_WUPE0(x)  
((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE0_SHIFT))&LLWU_PE1_WUPE0_MASK  
  
#define LLWU_PE1_WUPE1_MASK          0xCu  
#define LLWU_PE1_WUPE1_SHIFT         2  
#define LLWU_PE1_WUPE1_WIDTH         2  
#define LLWU_PE1_WUPE1(x)  
((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE1_SHIFT))&LLWU_PE1_WUPE1_MASK  
  
#define LLWU_PE1_WUPE2_MASK          0x30u  
#define LLWU_PE1_WUPE2_SHIFT         4  
#define LLWU_PE1_WUPE2_WIDTH         2  
#define LLWU_PE1_WUPE2(x)  
((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE2_SHIFT))&LLWU_PE1_WUPE2_MASK  
  
#define LLWU_PE1_WUPE3_MASK          0xC0u  
#define LLWU_PE1_WUPE3_SHIFT         6
```

```

#define LLWU_PE1_WUPE3_WIDTH          2
#define LLWU_PE1_WUPE3(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE3_SHIFT))&LLWU_PE1_WUPE3_MASK)
/* PE2 Bit Fields */

#define LLWU_PE2_WUPE4_MASK           0x3u
#define LLWU_PE2_WUPE4_SHIFT          0
#define LLWU_PE2_WUPE4_WIDTH          2
#define LLWU_PE2_WUPE4(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE4_SHIFT))&LLWU_PE2_WUPE4_MASK)
#define LLWU_PE2_WUPE5_MASK           0xCu
#define LLWU_PE2_WUPE5_SHIFT          2
#define LLWU_PE2_WUPE5_WIDTH          2
#define LLWU_PE2_WUPE5(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE5_SHIFT))&LLWU_PE2_WUPE5_MASK)
#define LLWU_PE2_WUPE6_MASK           0x30u
#define LLWU_PE2_WUPE6_SHIFT          4
#define LLWU_PE2_WUPE6_WIDTH          2
#define LLWU_PE2_WUPE6(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE6_SHIFT))&LLWU_PE2_WUPE6_MASK)
#define LLWU_PE2_WUPE7_MASK           0xC0u
#define LLWU_PE2_WUPE7_SHIFT          6
#define LLWU_PE2_WUPE7_WIDTH          2
#define LLWU_PE2_WUPE7(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE7_SHIFT))&LLWU_PE2_WUPE7_MASK)
/* PE3 Bit Fields */

#define LLWU_PE3_WUPE8_MASK           0x3u
#define LLWU_PE3_WUPE8_SHIFT          0
#define LLWU_PE3_WUPE8_WIDTH          2
#define LLWU_PE3_WUPE8(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE8_SHIFT))&LLWU_PE3_WUPE8_MASK)
#define LLWU_PE3_WUPE9_MASK           0xCu

```

```

#define LLWU_PE3_WUPE9_SHIFT          2
#define LLWU_PE3_WUPE9_WIDTH          2
#define LLWU_PE3_WUPE9(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE9_SHIFT))&LLWU_PE3_WUPE9_MASK)

#define LLWU_PE3_WUPE10_MASK          0x30u
#define LLWU_PE3_WUPE10_SHIFT         4
#define LLWU_PE3_WUPE10_WIDTH         2
#define LLWU_PE3_WUPE10(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE10_SHIFT))&LLWU_PE3_WUPE10_MASK)

#define LLWU_PE3_WUPE11_MASK          0xC0u
#define LLWU_PE3_WUPE11_SHIFT         6
#define LLWU_PE3_WUPE11_WIDTH         2
#define LLWU_PE3_WUPE11(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE11_SHIFT))&LLWU_PE3_WUPE11_MASK)

/* PE4 Bit Fields */

#define LLWU_PE4_WUPE12_MASK          0x3u
#define LLWU_PE4_WUPE12_SHIFT         0
#define LLWU_PE4_WUPE12_WIDTH         2
#define LLWU_PE4_WUPE12(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE12_SHIFT))&LLWU_PE4_WUPE12_MASK)

#define LLWU_PE4_WUPE13_MASK          0xCu
#define LLWU_PE4_WUPE13_SHIFT         2
#define LLWU_PE4_WUPE13_WIDTH         2
#define LLWU_PE4_WUPE13(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE13_SHIFT))&LLWU_PE4_WUPE13_MASK)

#define LLWU_PE4_WUPE14_MASK          0x30u
#define LLWU_PE4_WUPE14_SHIFT         4
#define LLWU_PE4_WUPE14_WIDTH         2
#define LLWU_PE4_WUPE14(x)
(((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE14_SHIFT))&LLWU_PE4_WUPE14_MASK)

#define LLWU_PE4_WUPE15_MASK          0xC0u

```

```

#define LLWU_PE4_WUPE15_SHIFT          6
#define LLWU_PE4_WUPE15_WIDTH          2
#define LLWU_PE4_WUPE15(x)             (((uint8_t)((uint8_t)(x)<<LLWU_PE4_WUPE15_SHIFT))&LLWU_PE4_WUPE15_MASK)

/* ME Bit Fields */

#define LLWU_ME_WUME0_MASK            0x1u
#define LLWU_ME_WUME0_SHIFT           0
#define LLWU_ME_WUME0_WIDTH           1
#define LLWU_ME_WUME0(x)              (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME0_SHIFT))&LLWU_ME_WUME0_MASK)

#define LLWU_ME_WUME1_MASK            0x2u
#define LLWU_ME_WUME1_SHIFT           1
#define LLWU_ME_WUME1_WIDTH           1
#define LLWU_ME_WUME1(x)              (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME1_SHIFT))&LLWU_ME_WUME1_MASK)

#define LLWU_ME_WUME2_MASK            0x4u
#define LLWU_ME_WUME2_SHIFT           2
#define LLWU_ME_WUME2_WIDTH           1
#define LLWU_ME_WUME2(x)              (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME2_SHIFT))&LLWU_ME_WUME2_MASK)

#define LLWU_ME_WUME3_MASK            0x8u
#define LLWU_ME_WUME3_SHIFT           3
#define LLWU_ME_WUME3_WIDTH           1
#define LLWU_ME_WUME3(x)              (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME3_SHIFT))&LLWU_ME_WUME3_MASK)

#define LLWU_ME_WUME4_MASK            0x10u
#define LLWU_ME_WUME4_SHIFT           4
#define LLWU_ME_WUME4_WIDTH           1
#define LLWU_ME_WUME4(x)              (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME4_SHIFT))&LLWU_ME_WUME4_MASK)

#define LLWU_ME_WUME5_MASK            0x20u

```

```

#define LLWU_ME_WUME5_SHIFT      5
#define LLWU_ME_WUME5_WIDTH      1
#define LLWU_ME_WUME5(x) (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME5_SHIFT))&LLWU_ME_WUME5_MASK)

#define LLWU_ME_WUME6_MASK      0x40u
#define LLWU_ME_WUME6_SHIFT      6
#define LLWU_ME_WUME6_WIDTH      1
#define LLWU_ME_WUME6(x) (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME6_SHIFT))&LLWU_ME_WUME6_MASK)

#define LLWU_ME_WUME7_MASK      0x80u
#define LLWU_ME_WUME7_SHIFT      7
#define LLWU_ME_WUME7_WIDTH      1
#define LLWU_ME_WUME7(x) (((uint8_t)((uint8_t)(x)<<LLWU_ME_WUME7_SHIFT))&LLWU_ME_WUME7_MASK)

/* F1 Bit Fields */

#define LLWU_F1_WUFO_MASK        0x1u
#define LLWU_F1_WUFO_SHIFT       0
#define LLWU_F1_WUFO_WIDTH       1
#define LLWU_F1_WUFO(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUFO_SHIFT))&LLWU_F1_WUFO_MASK)

#define LLWU_F1_WUF1_MASK        0x2u
#define LLWU_F1_WUF1_SHIFT       1
#define LLWU_F1_WUF1_WIDTH       1
#define LLWU_F1_WUF1(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF1_SHIFT))&LLWU_F1_WUF1_MASK)

#define LLWU_F1_WUF2_MASK        0x4u
#define LLWU_F1_WUF2_SHIFT       2
#define LLWU_F1_WUF2_WIDTH       1
#define LLWU_F1_WUF2(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF2_SHIFT))&LLWU_F1_WUF2_MASK)

#define LLWU_F1_WUF3_MASK        0x8u

```

```

#define LLWU_F1_WUF3_SHIFT          3
#define LLWU_F1_WUF3_WIDTH          1
#define LLWU_F1_WUF3(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF3_SHIFT))&LLWU_F1_WUF3_MASK)
#define LLWU_F1_WUF4_MASK           0x10u
#define LLWU_F1_WUF4_SHIFT          4
#define LLWU_F1_WUF4_WIDTH          1
#define LLWU_F1_WUF4(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF4_SHIFT))&LLWU_F1_WUF4_MASK)
#define LLWU_F1_WUF5_MASK           0x20u
#define LLWU_F1_WUF5_SHIFT          5
#define LLWU_F1_WUF5_WIDTH          1
#define LLWU_F1_WUF5(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF5_SHIFT))&LLWU_F1_WUF5_MASK)
#define LLWU_F1_WUF6_MASK           0x40u
#define LLWU_F1_WUF6_SHIFT          6
#define LLWU_F1_WUF6_WIDTH          1
#define LLWU_F1_WUF6(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF6_SHIFT))&LLWU_F1_WUF6_MASK)
#define LLWU_F1_WUF7_MASK           0x80u
#define LLWU_F1_WUF7_SHIFT          7
#define LLWU_F1_WUF7_WIDTH          1
#define LLWU_F1_WUF7(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF7_SHIFT))&LLWU_F1_WUF7_MASK)

/* F2 Bit Fields */

#define LLWU_F2_WUF8_MASK           0x1u
#define LLWU_F2_WUF8_SHIFT          0
#define LLWU_F2_WUF8_WIDTH          1
#define LLWU_F2_WUF8(x)             (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF8_SHIFT))&LLWU_F2_WUF8_MASK)
#define LLWU_F2_WUF9_MASK           0x2u

```

```
#define LLWU_F2_WUF9_SHIFT          1
#define LLWU_F2_WUF9_WIDTH           1
#define LLWU_F2_WUF9(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF9_SHIFT))&LLWU_F2_WUF9_MASK)

#define LLWU_F2_WUF10_MASK          0x4u
#define LLWU_F2_WUF10_SHIFT         2
#define LLWU_F2_WUF10_WIDTH         1
#define LLWU_F2_WUF10(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF10_SHIFT))&LLWU_F2_WUF10_MASK)

#define LLWU_F2_WUF11_MASK          0x8u
#define LLWU_F2_WUF11_SHIFT         3
#define LLWU_F2_WUF11_WIDTH         1
#define LLWU_F2_WUF11(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF11_SHIFT))&LLWU_F2_WUF11_MASK)

#define LLWU_F2_WUF12_MASK          0x10u
#define LLWU_F2_WUF12_SHIFT         4
#define LLWU_F2_WUF12_WIDTH         1
#define LLWU_F2_WUF12(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF12_SHIFT))&LLWU_F2_WUF12_MASK)

#define LLWU_F2_WUF13_MASK          0x20u
#define LLWU_F2_WUF13_SHIFT         5
#define LLWU_F2_WUF13_WIDTH         1
#define LLWU_F2_WUF13(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF13_SHIFT))&LLWU_F2_WUF13_MASK)

#define LLWU_F2_WUF14_MASK          0x40u
#define LLWU_F2_WUF14_SHIFT         6
#define LLWU_F2_WUF14_WIDTH         1
#define LLWU_F2_WUF14(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF14_SHIFT))&LLWU_F2_WUF14_MASK)

#define LLWU_F2_WUF15_MASK          0x80u
#define LLWU_F2_WUF15_SHIFT         7
```

```

#define LLWU_F2_WUF15_WIDTH          1
#define LLWU_F2_WUF15(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF15_SHIFT))&LLWU_F2_WUF15_MASK)

/* F3 Bit Fields */

#define LLWU_F3_MWUF0_MASK           0x1u
#define LLWU_F3_MWUF0_SHIFT          0
#define LLWU_F3_MWUF0_WIDTH          1
#define LLWU_F3_MWUF0(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF0_SHIFT))&LLWU_F3_MWUF0_MASK)

#define LLWU_F3_MWUF1_MASK           0x2u
#define LLWU_F3_MWUF1_SHIFT          1
#define LLWU_F3_MWUF1_WIDTH          1
#define LLWU_F3_MWUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF1_SHIFT))&LLWU_F3_MWUF1_MASK)

#define LLWU_F3_MWUF2_MASK           0x4u
#define LLWU_F3_MWUF2_SHIFT          2
#define LLWU_F3_MWUF2_WIDTH          1
#define LLWU_F3_MWUF2(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF2_SHIFT))&LLWU_F3_MWUF2_MASK)

#define LLWU_F3_MWUF3_MASK           0x8u
#define LLWU_F3_MWUF3_SHIFT          3
#define LLWU_F3_MWUF3_WIDTH          1
#define LLWU_F3_MWUF3(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF3_SHIFT))&LLWU_F3_MWUF3_MASK)

#define LLWU_F3_MWUF4_MASK           0x10u
#define LLWU_F3_MWUF4_SHIFT          4
#define LLWU_F3_MWUF4_WIDTH          1
#define LLWU_F3_MWUF4(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF4_SHIFT))&LLWU_F3_MWUF4_MASK)

#define LLWU_F3_MWUF5_MASK           0x20u
#define LLWU_F3_MWUF5_SHIFT          5

```

```

#define LLWU_F3_MWUF5_WIDTH          1
#define LLWU_F3_MWUF5(x)             (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF5_SHIFT))&LLWU_F3_MWUF5_MASK)

#define LLWU_F3_MWUF6_MASK           0x40u
#define LLWU_F3_MWUF6_SHIFT          6
#define LLWU_F3_MWUF6_WIDTH          1
#define LLWU_F3_MWUF6(x)             (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF6_SHIFT))&LLWU_F3_MWUF6_MASK)

#define LLWU_F3_MWUF7_MASK           0x80u
#define LLWU_F3_MWUF7_SHIFT          7
#define LLWU_F3_MWUF7_WIDTH          1
#define LLWU_F3_MWUF7(x)             (((uint8_t)((uint8_t)(x)<<LLWU_F3_MWUF7_SHIFT))&LLWU_F3_MWUF7_MASK)

/* FILT1 Bit Fields */

#define LLWU_FILT1_FILTSEL_MASK      0xFu
#define LLWU_FILT1_FILTSEL_SHIFT     0
#define LLWU_FILT1_FILTSEL_WIDTH     4
#define LLWU_FILT1_FILTSEL(x)        (((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTSEL_SHIFT))&LLWU_FILT1_FILTSEL_MASK)

#define LLWU_FILT1_FILTE_MASK         0x60u
#define LLWU_FILT1_FILTE_SHIFT        5
#define LLWU_FILT1_FILTE_WIDTH        2
#define LLWU_FILT1_FILTE(x)          (((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTE_SHIFT))&LLWU_FILT1_FILTE_MASK)

#define LLWU_FILT1_FILTF_MASK         0x80u
#define LLWU_FILT1_FILTF_SHIFT        7
#define LLWU_FILT1_FILTF_WIDTH        1
#define LLWU_FILT1_FILTF(x)          (((uint8_t)((uint8_t)(x)<<LLWU_FILT1_FILTF_SHIFT))&LLWU_FILT1_FILTF_MASK)

/* FILT2 Bit Fields */

#define LLWU_FILT2_FILTSEL_MASK      0xFu

```

```

#define LLWU_FILT2_FILTSEL_SHIFT      0
#define LLWU_FILT2_FILTSEL_WIDTH      4
#define LLWU_FILT2_FILTSEL(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTSEL_SHIFT))&LLWU_FILT2_FILTSEL_MASK)

#define LLWU_FILT2_FILTE_MASK         0x60u
#define LLWU_FILT2_FILTE_SHIFT        5
#define LLWU_FILT2_FILTE_WIDTH        2
#define LLWU_FILT2_FILTE(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTE_SHIFT))&LLWU_FILT2_FILTE_MASK)

#define LLWU_FILT2_FILTF_MASK         0x80u
#define LLWU_FILT2_FILTF_SHIFT        7
#define LLWU_FILT2_FILTF_WIDTH        1
#define LLWU_FILT2_FILTF(x)
(((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTF_SHIFT))&LLWU_FILT2_FILTF_MASK)

```

```

/*
 * @}
 */
/* /* end of group LLWU_Register_Masks */

```

```

/* LLWU - Peripheral instance base addresses */
/** Peripheral LLWU base address */
#define LLWU_BASE                  (0x4007C000u)
/** Peripheral LLWU base pointer */
#define LLWU                      ((LLWU_Type *)LLWU_BASE)
#define LLWU_BASE_PTR              (LLWU)
/** Array initializer of LLWU peripheral base addresses */
#define LLWU_BASE_ADDRS            { LLWU_BASE }
/** Array initializer of LLWU peripheral base pointers */
#define LLWU_BASE_PTRS             { LLWU }

```

```

/*
-----  

-- LLWU - Register accessor macros  

----- */
  

/*!  

 * @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros  

 * @{
 * /  

  

/* LLWU - Register instance definitions */  

/* LLWU */  

#define LLWU_PE1           LLWU_PE1_REG(LLWU)  

#define LLWU_PE2           LLWU_PE2_REG(LLWU)  

#define LLWU_PE3           LLWU_PE3_REG(LLWU)  

#define LLWU_PE4           LLWU_PE4_REG(LLWU)  

#define LLWU_ME            LLWU_ME_REG(LLWU)  

#define LLWU_F1             LLWU_F1_REG(LLWU)  

#define LLWU_F2             LLWU_F2_REG(LLWU)  

#define LLWU_F3             LLWU_F3_REG(LLWU)  

#define LLWU_FILTER1        LLWU_FILTER1_REG(LLWU)  

#define LLWU_FILTER2        LLWU_FILTER2_REG(LLWU)  

  

/*!  

 * @}
 * /* end of group LLWU_Register_Accessor_Macros */

```

```

/*!
* @}

/* /* end of group LLWU_Peripheral_Access_Layer */

/*
-----*
-- LPTMR Peripheral Access Layer
----- */
/*!

* @addtogroup LPTMR_Peripheral_Access_Layer LPTMR Peripheral Access Layer
* @{

*/
/** LPTMR - Register Layout Typedef */

typedef struct {

    __IO uint32_t CSR;           /**< Low Power Timer Control Status Register, offset: 0x0 */
    __IO uint32_t PSR;           /**< Low Power Timer Prescale Register, offset: 0x4 */
    __IO uint32_t CMR;           /**< Low Power Timer Compare Register, offset: 0x8 */
    __IO uint32_t CNR;           /**< Low Power Timer Counter Register, offset: 0xC */

} LPTMR_Type, *LPTMR_MemMapPtr;

/*
-----*
-- LPTMR - Register accessor macros
----- */
/*!

* @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
* @{

```

```
 */
```

```
/* LPTMR - Register accessors */

#define LPTMR_CSR_REG(base)      ((base)->CSR)
#define LPTMR_PSR_REG(base)      ((base)->PSR)
#define LPTMR_CMR_REG(base)      ((base)->CMR)
#define LPTMR_CNR_REG(base)      ((base)->CNR)
```

```
/*!
```

```
* @}
```

```
/* /* end of group LPTMR_Register_Accessor_Macros */
```

```
/* -----
```

```
-- LPTMR Register Masks
```

```
----- */
```

```
/*!
```

```
* @addtogroup LPTMR_Register_Masks LPTMR Register Masks
```

```
* @{
```

```
*/
```

```
/* CSR Bit Fields */
```

```
#define LPTMR_CSR_TEN_MASK      0x1u
#define LPTMR_CSR_TEN_SHIFT      0
#define LPTMR_CSR_TEN_WIDTH      1

#define LPTMR_CSR_TEN(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR_TEN_SHIFT))&LPTMR_CSR_TEN_MASK)
```

```

#define LPTMR_CSR_TMS_MASK          0x2u
#define LPTMR_CSR_TMS_SHIFT         1
#define LPTMR_CSR_TMS_WIDTH         1
#define LPTMR_CSR_TMS(x) (((uint32_t)(x)<<LPTMR_CSR_TMS_SHIFT))&LPTMR_CSR_TMS_MASK)

#define LPTMR_CSR_TFC_MASK          0x4u
#define LPTMR_CSR_TFC_SHIFT         2
#define LPTMR_CSR_TFC_WIDTH         1
#define LPTMR_CSR_TFC(x) (((uint32_t)(x)<<LPTMR_CSR_TFC_SHIFT))&LPTMR_CSR_TFC_MASK)

#define LPTMR_CSR TPP_MASK          0x8u
#define LPTMR_CSR TPP_SHIFT         3
#define LPTMR_CSR TPP_WIDTH         1
#define LPTMR_CSR TPP(x) (((uint32_t)(x)<<LPTMR_CSR TPP_SHIFT))&LPTMR_CSR TPP_MASK)

#define LPTMR_CSR TPS_MASK          0x30u
#define LPTMR_CSR TPS_SHIFT         4
#define LPTMR_CSR TPS_WIDTH         2
#define LPTMR_CSR TPS(x) (((uint32_t)(x)<<LPTMR_CSR TPS_SHIFT))&LPTMR_CSR TPS_MASK)

#define LPTMR_CSR TIE_MASK          0x40u
#define LPTMR_CSR TIE_SHIFT         6
#define LPTMR_CSR TIE_WIDTH         1
#define LPTMR_CSR TIE(x) (((uint32_t)(x)<<LPTMR_CSR TIE_SHIFT))&LPTMR_CSR TIE_MASK)

#define LPTMR_CSR TCF_MASK          0x80u
#define LPTMR_CSR TCF_SHIFT         7
#define LPTMR_CSR TCF_WIDTH         1
#define LPTMR_CSR TCF(x) (((uint32_t)(x)<<LPTMR_CSR TCF_SHIFT))&LPTMR_CSR TCF_MASK)

/* PSR Bit Fields */

```

```

#define LPTMR_PSR_PCS_MASK           0x3u
#define LPTMR_PSR_PCS_SHIFT          0
#define LPTMR_PSR_PCS_WIDTH          2
#define LPTMR_PSR_PCS(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PCS_SHIFT))&LPTMR_PSR_PCS_MASK)
#define LPTMR_PSR_PBYP_MASK          0x4u
#define LPTMR_PSR_PBYP_SHIFT          2
#define LPTMR_PSR_PBYP_WIDTH          1
#define LPTMR_PSR_PBYP(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PBYP_SHIFT))&LPTMR_PSR_PBYP_MASK)
#define LPTMR_PSR_PRESCALE_MASK       0x78u
#define LPTMR_PSR_PRESCALE_SHIFT      3
#define LPTMR_PSR_PRESCALE_WIDTH      4
#define LPTMR_PSR_PRESCALE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR_PRESCALE_SHIFT))&LPTMR_PSR_PRESCALE_MASK)
/* CMR Bit Fields */
#define LPTMR_CMR_COMPARE_MASK        0xFFFFu
#define LPTMR_CMR_COMPARE_SHIFT       0
#define LPTMR_CMR_COMPARE_WIDTH       16
#define LPTMR_CMR_COMPARE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CMR_COMPARE_SHIFT))&LPTMR_CMR_COMPARE_MASK)
/* CNR Bit Fields */
#define LPTMR_CNR_COUNTER_MASK        0xFFFFu
#define LPTMR_CNR_COUNTER_SHIFT       0
#define LPTMR_CNR_COUNTER_WIDTH       16
#define LPTMR_CNR_COUNTER(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CNR_COUNTER_SHIFT))&LPTMR_CNR_COUNTER_MASK)

*/
* @}

```

```
/* /* end of group LPTMR_Register_Masks */
```

```
/* LPTMR - Peripheral instance base addresses */

/** Peripheral LPTMRO base address */

#define LPTMRO_BASE          (0x40040000u)

/** Peripheral LPTMRO base pointer */

#define LPTMRO           ((LPTMR_Type *)LPTMRO_BASE)

#define LPTMRO_BASE_PTR      (LPTMRO)

/** Array initializer of LPTMR peripheral base addresses */

#define LPTMR_BASE_ADDRS     { LPTMRO_BASE }

/** Array initializer of LPTMR peripheral base pointers */

#define LPTMR_BASE_PTRS       { LPTMRO }
```

```
/*
-----*
-- LPTMR - Register accessor macros
----- */
```

```
/*!

* @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
* @{
*/

```

```
/* LPTMR - Register instance definitions */

/* LPTMRO */

#define LPTMRO_CSR          LPTMR_CSR_REG(LPTMRO)
#define LPTMRO_PSR          LPTMR_PSR_REG(LPTMRO)
#define LPTMRO_CMR          LPTMR_CMR_REG(LPTMRO)
```

```
#define LPTMR0_CNR           LPTMR_CNR_REG(LPTMR0)
```

```
/*!  
 * @}  
 */ /* end of group LPTMR_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group LPTMR_Peripheral_Access_Layer */
```

```
/* -----  
-- MCG Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup MCG_Peripheral_Access_Layer MCG Peripheral Access Layer  
 * @{  
 */
```

```
/** MCG - Register Layout Typedef */  
  
typedef struct {  
    __IO uint8_t C1;          /**< MCG Control 1 Register, offset: 0x0 */  
    __IO uint8_t C2;          /**< MCG Control 2 Register, offset: 0x1 */  
    __IO uint8_t C3;          /**< MCG Control 3 Register, offset: 0x2 */  
    __IO uint8_t C4;          /**< MCG Control 4 Register, offset: 0x3 */  
    __IO uint8_t C5;          /**< MCG Control 5 Register, offset: 0x4 */  
    __IO uint8_t C6;          /**< MCG Control 6 Register, offset: 0x5 */
```

```

__IO uint8_t S;                                /**< MCG Status Register, offset: 0x6 */

    uint8_t RESERVED_0[1];

__IO uint8_t SC;                               /**< MCG Status and Control Register, offset: 0x8 */

    uint8_t RESERVED_1[1];

__IO uint8_t ATCVH;                            /**< MCG Auto Trim Compare Value High Register, offset: 0xA */

__IO uint8_t ATCVL;                            /**< MCG Auto Trim Compare Value Low Register, offset: 0xB */

__I uint8_t C7;                               /**< MCG Control 7 Register, offset: 0xC */

__IO uint8_t C8;                               /**< MCG Control 8 Register, offset: 0xD */

__I uint8_t C9;                               /**< MCG Control 9 Register, offset: 0xE */

__I uint8_t C10;                             /**< MCG Control 10 Register, offset: 0xF */

} MCG_Type, *MCG_MemMapPtr;

```

```

/*
-----*
-- MCG - Register accessor macros
----- */

```

```

/*!
* @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros
* @{
*/

```

```

/* MCG - Register accessors */

#define MCG_C1_REG(base)      ((base)->C1)
#define MCG_C2_REG(base)      ((base)->C2)
#define MCG_C3_REG(base)      ((base)->C3)
#define MCG_C4_REG(base)      ((base)->C4)
#define MCG_C5_REG(base)      ((base)->C5)
#define MCG_C6_REG(base)      ((base)->C6)

```

```
#define MCG_S_REG(base) ((base)->S)
#define MCG_SC_REG(base) ((base)->SC)
#define MCG_ATCVH_REG(base) ((base)->ATCVH)
#define MCG_ATCVL_REG(base) ((base)->ATCVL)
#define MCG_C7_REG(base) ((base)->C7)
#define MCG_C8_REG(base) ((base)->C8)
#define MCG_C9_REG(base) ((base)->C9)
#define MCG_C10_REG(base) ((base)->C10)
```

```
/*!
 * @}
 */
/* /* end of group MCG_Register_Accessor_Macros */
```

```
/*
-----*
-- MCG Register Masks
----- */
```

```
/*
* @addtogroup MCG_Register_Masks MCG Register Masks
* @{
*/

```

```
/* C1 Bit Fields */

#define MCG_C1_IREFSTEN_MASK 0x1u
#define MCG_C1_IREFSTEN_SHIFT 0
#define MCG_C1_IREFSTEN_WIDTH 1

#define MCG_C1_IREFSTEN(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_IREFSTEN_SHIFT))&MCG_C1_IREFSTEN_MASK)
```

```

#define MCG_C1_IRCLKEN_MASK          0x2u
#define MCG_C1_IRCLKEN_SHIFT         1
#define MCG_C1_IRCLKEN_WIDTH         1
#define MCG_C1_IRCLKEN(x)            (((uint8_t)((uint8_t)(x)<<MCG_C1_IRCLKEN_SHIFT))&MCG_C1_IRCLKEN_MASK)

#define MCG_C1_IREFS_MASK            0x4u
#define MCG_C1_IREFS_SHIFT           2
#define MCG_C1_IREFS_WIDTH           1
#define MCG_C1_IREFS(x)              (((uint8_t)((uint8_t)(x)<<MCG_C1_IREFS_SHIFT))&MCG_C1_IREFS_MASK)

#define MCG_C1_FRDIV_MASK            0x38u
#define MCG_C1_FRDIV_SHIFT           3
#define MCG_C1_FRDIV_WIDTH           3
#define MCG_C1_FRDIV(x)              (((uint8_t)((uint8_t)(x)<<MCG_C1_FRDIV_SHIFT))&MCG_C1_FRDIV_MASK)

#define MCG_C1_CLKS_MASK              0xC0u
#define MCG_C1_CLKS_SHIFT             6
#define MCG_C1_CLKS_WIDTH             2
#define MCG_C1_CLKS(x)                (((uint8_t)((uint8_t)(x)<<MCG_C1_CLKS_SHIFT))&MCG_C1_CLKS_MASK)

/* C2 Bit Fields */

#define MCG_C2_IRCS_MASK             0x1u
#define MCG_C2_IRCS_SHIFT             0
#define MCG_C2_IRCS_WIDTH             1
#define MCG_C2_IRCS(x)                (((uint8_t)((uint8_t)(x)<<MCG_C2_IRCS_SHIFT))&MCG_C2_IRCS_MASK)

#define MCG_C2_LP_MASK                0x2u
#define MCG_C2_LP_SHIFT               1
#define MCG_C2_LP_WIDTH               1
#define MCG_C2_LP(x)                  (((uint8_t)((uint8_t)(x)<<MCG_C2_LP_SHIFT))&MCG_C2_LP_MASK)

```

```

#define MCG_C2_EREFS0_MASK          0x4u
#define MCG_C2_EREFS0_SHIFT         2
#define MCG_C2_EREFS0_WIDTH         1
#define MCG_C2_EREFS0(x)            (((uint8_t)((uint8_t)(x)<<MCG_C2_EREFS0_SHIFT))&MCG_C2_EREFS0_MASK)

#define MCG_C2_HGO0_MASK           0x8u
#define MCG_C2_HGO0_SHIFT          3
#define MCG_C2_HGO0_WIDTH          1
#define MCG_C2_HGO0(x)             (((uint8_t)((uint8_t)(x)<<MCG_C2_HGO0_SHIFT))&MCG_C2_HGO0_MASK)

#define MCG_C2_RANGE0_MASK          0x30u
#define MCG_C2_RANGE0_SHIFT         4
#define MCG_C2_RANGE0_WIDTH         2
#define MCG_C2_RANGE0(x)            (((uint8_t)((uint8_t)(x)<<MCG_C2_RANGE0_SHIFT))&MCG_C2_RANGE0_MASK)

#define MCG_C2_LOCRE0_MASK          0x80u
#define MCG_C2_LOCRE0_SHIFT         7
#define MCG_C2_LOCRE0_WIDTH         1
#define MCG_C2_LOCRE0(x)            (((uint8_t)((uint8_t)(x)<<MCG_C2_LOCRE0_SHIFT))&MCG_C2_LOCRE0_MASK)

/* C3 Bit Fields */

#define MCG_C3_SCTRIM_MASK          0xFFu
#define MCG_C3_SCTRIM_SHIFT         0
#define MCG_C3_SCTRIM_WIDTH         8
#define MCG_C3_SCTRIM(x)            (((uint8_t)((uint8_t)(x)<<MCG_C3_SCTRIM_SHIFT))&MCG_C3_SCTRIM_MASK)

/* C4 Bit Fields */

#define MCG_C4_SCFTRIM_MASK          0x1u
#define MCG_C4_SCFTRIM_SHIFT         0
#define MCG_C4_SCFTRIM_WIDTH         1

```

```

#define MCG_C4_SCFTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_SCFTRIM_SHIFT))&MCG_C4_SCFTRIM_MASK)

#define MCG_C4_FCTRIM_MASK          0x1Eu
#define MCG_C4_FCTRIM_SHIFT        1
#define MCG_C4_FCTRIM_WIDTH        4

#define MCG_C4_FCTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_FCTRIM_SHIFT))&MCG_C4_FCTRIM_MASK)

#define MCG_C4_DRST_DRS_MASK        0x60u
#define MCG_C4_DRST_DRS_SHIFT      5
#define MCG_C4_DRST_DRS_WIDTH      2

#define MCG_C4_DRST_DRS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_DRST_DRS_SHIFT))&MCG_C4_DRST_DRS_MASK)

#define MCG_C4_DMX32_MASK           0x80u
#define MCG_C4_DMX32_SHIFT          7
#define MCG_C4_DMX32_WIDTH          1

#define MCG_C4_DMX32(x)
(((uint8_t)((uint8_t)(x)<<MCG_C4_DMX32_SHIFT))&MCG_C4_DMX32_MASK)

/* C5 Bit Fields */

#define MCG_C5_PRDIV0_MASK          0x1Fu
#define MCG_C5_PRDIV0_SHIFT         0
#define MCG_C5_PRDIV0_WIDTH         5

#define MCG_C5_PRDIV0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C5_PRDIV0_SHIFT))&MCG_C5_PRDIV0_MASK)

#define MCG_C5_PLLSTENO_MASK         0x20u
#define MCG_C5_PLLSTENO_SHIFT       5
#define MCG_C5_PLLSTENO_WIDTH       1

#define MCG_C5_PLLSTENO(x)
(((uint8_t)((uint8_t)(x)<<MCG_C5_PLLSTENO_SHIFT))&MCG_C5_PLLSTENO_MASK)

#define MCG_C5_PLLCLKENO_MASK        0x40u
#define MCG_C5_PLLCLKENO_SHIFT      6
#define MCG_C5_PLLCLKENO_WIDTH      1

```

```

#define MCG_C5_PLLCLKENO(x)
(((uint8_t)((uint8_t)(x)<<MCG_C5_PLLCLKENO_SHIFT))&MCG_C5_PLLCLKENO_MASK)

/* C6 Bit Fields */

#define MCG_C6_VDIV0_MASK          0x1Fu
#define MCG_C6_VDIV0_SHIFT         0
#define MCG_C6_VDIV0_WIDTH         5
#define MCG_C6_VDIV0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_VDIV0_SHIFT))&MCG_C6_VDIV0_MASK)

#define MCG_C6_CME0_MASK          0x20u
#define MCG_C6_CME0_SHIFT         5
#define MCG_C6_CME0_WIDTH         1
#define MCG_C6_CME0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_CME0_SHIFT))&MCG_C6_CME0_MASK)

#define MCG_C6_PLLS_MASK          0x40u
#define MCG_C6_PLLS_SHIFT         6
#define MCG_C6_PLLS_WIDTH         1
#define MCG_C6_PLLS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_PLLS_SHIFT))&MCG_C6_PLLS_MASK)

#define MCG_C6_LOLIE0_MASK         0x80u
#define MCG_C6_LOLIE0_SHIFT        7
#define MCG_C6_LOLIE0_WIDTH        1
#define MCG_C6_LOLIE0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_LOLIE0_SHIFT))&MCG_C6_LOLIE0_MASK)

/* S Bit Fields */

#define MCG_S_IRCST_MASK          0x1u
#define MCG_S_IRCST_SHIFT         0
#define MCG_S_IRCST_WIDTH         1
#define MCG_S_IRCST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_IRCST_SHIFT))&MCG_S_IRCST_MASK)

#define MCG_S_OSCINIT0_MASK        0x2u
#define MCG_S_OSCINIT0_SHIFT       1

```

```

#define MCG_S_OSCINIT0_WIDTH          1
#define MCG_S_OSCINIT0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_OSCINIT0_SHIFT))&MCG_S_OSCINIT0_MASK)

#define MCG_S_CLKST_MASK            0xCu
#define MCG_S_CLKST_SHIFT           2
#define MCG_S_CLKST_WIDTH           2
#define MCG_S_CLKST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_CLKST_SHIFT))&MCG_S_CLKST_MASK)

#define MCG_S_IREFST_MASK           0x10u
#define MCG_S_IREFST_SHIFT          4
#define MCG_S_IREFST_WIDTH          1
#define MCG_S_IREFST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_IREFST_SHIFT))&MCG_S_IREFST_MASK)

#define MCG_S_PLLST_MASK            0x20u
#define MCG_S_PLLST_SHIFT           5
#define MCG_S_PLLST_WIDTH           1
#define MCG_S_PLLST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_PLLST_SHIFT))&MCG_S_PLLST_MASK)

#define MCG_S_LOCK0_MASK             0x40u
#define MCG_S_LOCK0_SHIFT            6
#define MCG_S_LOCK0_WIDTH            1
#define MCG_S_LOCK0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_LOCK0_SHIFT))&MCG_S_LOCK0_MASK)

#define MCG_S_LOLS0_MASK             0x80u
#define MCG_S_LOLS0_SHIFT            7
#define MCG_S_LOLS0_WIDTH            1
#define MCG_S_LOLS0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_LOLS0_SHIFT))&MCG_S_LOLS0_MASK)

/* SC Bit Fields */

#define MCG_SC_LOCS0_MASK           0x1u
#define MCG_SC_LOCS0_SHIFT           0

```

```

#define MCG_SC_LOCS0_WIDTH          1
#define MCG_SC_LOCS0(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_LOCS0_SHIFT))&MCG_SC_LOCS0_MASK)
#define MCG_SC_FCRDIV_MASK          0xEu
#define MCG_SC_FCRDIV_SHIFT         1
#define MCG_SC_FCRDIV_WIDTH         3
#define MCG_SC_FCRDIV(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_FCRDIV_SHIFT))&MCG_SC_FCRDIV_MASK)
#define MCG_SC_FLTPRSRV_MASK        0x10u
#define MCG_SC_FLTPRSRV_SHIFT       4
#define MCG_SC_FLTPRSRV_WIDTH       1
#define MCG_SC_FLTPRSRV(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_FLTPRSRV_SHIFT))&MCG_SC_FLTPRSRV_MASK)
#define MCG_SC_ATMF_MASK            0x20u
#define MCG_SC_ATMF_SHIFT           5
#define MCG_SC_ATMF_WIDTH           1
#define MCG_SC_ATMF(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMF_SHIFT))&MCG_SC_ATMF_MASK)
#define MCG_SC_ATMS_MASK            0x40u
#define MCG_SC_ATMS_SHIFT           6
#define MCG_SC_ATMS_WIDTH           1
#define MCG_SC_ATMS(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMS_SHIFT))&MCG_SC_ATMS_MASK)
#define MCG_SC_ATME_MASK            0x80u
#define MCG_SC_ATME_SHIFT           7
#define MCG_SC_ATME_WIDTH           1
#define MCG_SC_ATME(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATME_SHIFT))&MCG_SC_ATME_MASK)
/* ATCVH Bit Fields */
#define MCG_ATCVH_ATCVH_MASK         0xFFu
#define MCG_ATCVH_ATCVH_SHIFT        0

```

```

#define MCG_ATCVH_ATCVH_WIDTH          8
#define MCG_ATCVH_ATCVH(x) (((uint8_t)((uint8_t)(x)<<MCG_ATCVH_ATCVH_SHIFT))&MCG_ATCVH_ATCVH_MASK)
/* ATCVL Bit Fields */
#define MCG_ATCVL_ATCVL_MASK           0xFFu
#define MCG_ATCVL_ATCVL_SHIFT          0
#define MCG_ATCVL_ATCVL_WIDTH          8
#define MCG_ATCVL_ATCVL(x) (((uint8_t)((uint8_t)(x)<<MCG_ATCVL_ATCVL_SHIFT))&MCG_ATCVL_ATCVL_MASK)
/* C8 Bit Fields */
#define MCG_C8_LOLRE_MASK              0x40u
#define MCG_C8_LOLRE_SHIFT             6
#define MCG_C8_LOLRE_WIDTH             1
#define MCG_C8_LOLRE(x) (((uint8_t)((uint8_t)(x)<<MCG_C8_LOLRE_SHIFT))&MCG_C8_LOLRE_MASK)

/*
 * @}
 */

/*! /* end of group MCG_Register_Masks */

```

```

/* MCG - Peripheral instance base addresses */
/** Peripheral MCG base address */
#define MCG_BASE                      (0x40064000u)
/** Peripheral MCG base pointer */
#define MCG                         ((MCG_Type *)MCG_BASE)
#define MCG_BASE_PTR                  (MCG)
/** Array initializer of MCG peripheral base addresses */
#define MCG_BASE_ADDRS                { MCG_BASE }
/** Array initializer of MCG peripheral base pointers */

```

```

#define MCG_BASE_PTRS          { MCG }

/*
-----  

-- MCG - Register accessor macros  

----- */
/*!  

* @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros  

* @{
* /  

  

/* MCG - Register instance definitions */  

/* MCG */  

#define MCG_C1                  MCG_C1_REG(MCG)  

#define MCG_C2                  MCG_C2_REG(MCG)  

#define MCG_C3                  MCG_C3_REG(MCG)  

#define MCG_C4                  MCG_C4_REG(MCG)  

#define MCG_C5                  MCG_C5_REG(MCG)  

#define MCG_C6                  MCG_C6_REG(MCG)  

#define MCG_S                   MCG_S_REG(MCG)  

#define MCG_SC                 MCG_SC_REG(MCG)  

#define MCG_ATCVH              MCG_ATCVH_REG(MCG)  

#define MCG_ATCVL              MCG_ATCVL_REG(MCG)  

#define MCG_C7                  MCG_C7_REG(MCG)  

#define MCG_C8                  MCG_C8_REG(MCG)  

#define MCG_C9                  MCG_C9_REG(MCG)  

#define MCG_C10                 MCG_C10_REG(MCG)

```

```

/*!
* @}

/* /* end of group MCG_Register_Accessor_Macros */

/* MCG C2[EREFS] backward compatibility */

#define MCG_C2_EREFS_MASK      (MCG_C2_EREFS0_MASK)
#define MCG_C2_EREFS_SHIFT     (MCG_C2_EREFS0_SHIFT)
#define MCG_C2_EREFS_WIDTH     (MCG_C2_EREFS0_WIDTH)
#define MCG_C2_EREFS(x)        (MCG_C2_EREFS0(x))

/* MCG C2[HGO] backward compatibility */

#define MCG_C2_HGO_MASK       (MCG_C2_HGO0_MASK)
#define MCG_C2_HGO_SHIFT      (MCG_C2_HGO0_SHIFT)
#define MCG_C2_HGO_WIDTH      (MCG_C2_HGO0_WIDTH)
#define MCG_C2_HGO(x)         (MCG_C2_HGO0(x))

/* MCG C2[RANGE] backward compatibility */

#define MCG_C2_RANGE_MASK     (MCG_C2_RANGE0_MASK)
#define MCG_C2_RANGE_SHIFT    (MCG_C2_RANGE0_SHIFT)
#define MCG_C2_RANGE_WIDTH    (MCG_C2_RANGE0_WIDTH)
#define MCG_C2_RANGE(x)       (MCG_C2_RANGE0(x))

/*!
* @}

/* /* end of group MCG_Peripheral_Access_Layer */

/*
-----
```

```

-- MCM Peripheral Access Layer
----- */
/*!
* @addtogroup MCM_Peripheral_Access_Layer MCM Peripheral Access Layer
* @{
*/
/** MCM - Register Layout Typedef */
typedef struct {
    uint8_t RESERVED_0[8];
    __I uint16_t PLASC;           /**< Crossbar Switch (AXBS) Slave Configuration, offset: 0x8 */
    __I uint16_t PLAMC;           /**< Crossbar Switch (AXBS) Master Configuration, offset: 0xA */
    __IO uint32_t PLACR;          /**< Platform Control Register, offset: 0xC */
    uint8_t RESERVED_1[48];
    __IO uint32_t CPO;            /**< Compute Operation Control Register, offset: 0x40 */
} MCM_Type, *MCM_MemMapPtr;

/* -----
-- MCM - Register accessor macros
----- */

/*!
* @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros
* @{
*/
/* MCM - Register accessors */

```

```
#define MCM_PLASC_REG(base) ((base)->PLASC)
#define MCM_PLAMC_REG(base) ((base)->PLAMC)
#define MCM_PLACR_REG(base) ((base)->PLACR)
#define MCM_CPO_REG(base) ((base)->CPO)
```

```
/*
 * @}
 */
/* /* end of group MCM_Register_Accessor_Macros */
```

```
/*
-----  
-- MCM Register Masks  
----- */
```

```
/*
 * @addtogroup MCM_Register_Masks MCM Register Masks
 * @{
 */

```

```
/* PLASC Bit Fields */
#define MCM_PLASC_ASC_MASK 0xFFu
#define MCM_PLASC_ASC_SHIFT 0
#define MCM_PLASC_ASC_WIDTH 8
#define MCM_PLASC_ASC(x) (((uint16_t)((uint16_t)(x))<<MCM_PLASC_ASC_SHIFT))&MCM_PLASC_ASC_MASK)
```

```
/* PLAMC Bit Fields */
#define MCM_PLAMC_AMC_MASK 0xFFu
#define MCM_PLAMC_AMC_SHIFT 0
#define MCM_PLAMC_AMC_WIDTH 8
```

```

#define MCM_PLAMC_AMC(x)
(((uint16_t)((uint16_t)(x)<<MCM_PLAMC_AMC_SHIFT))&MCM_PLAMC_AMC_MASK)

/* PLACR Bit Fields */

#define MCM_PLACR_ARB_MASK          0x200u
#define MCM_PLACR_ARB_SHIFT         9
#define MCM_PLACR_ARB_WIDTH         1

#define MCM_PLACR_ARB(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_ARB_SHIFT))&MCM_PLACR_ARB_MASK)

#define MCM_PLACR_CFCC_MASK         0x400u
#define MCM_PLACR_CFCC_SHIFT        10
#define MCM_PLACR_CFCC_WIDTH        1

#define MCM_PLACR_CFCC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_CFCC_SHIFT))&MCM_PLACR_CFCC_MASK)

#define MCM_PLACR_DFCDA_MASK        0x800u
#define MCM_PLACR_DFCDA_SHIFT       11
#define MCM_PLACR_DFCDA_WIDTH       1

#define MCM_PLACR_DFCDA(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCDA_SHIFT))&MCM_PLACR_DFCDA_MASK)

#define MCM_PLACR_DFCIC_MASK        0x1000u
#define MCM_PLACR_DFCIC_SHIFT       12
#define MCM_PLACR_DFCIC_WIDTH       1

#define MCM_PLACR_DFCIC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCIC_SHIFT))&MCM_PLACR_DFCIC_MASK)

#define MCM_PLACR_DFCC_MASK         0x2000u
#define MCM_PLACR_DFCC_SHIFT        13
#define MCM_PLACR_DFCC_WIDTH        1

#define MCM_PLACR_DFCC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCC_SHIFT))&MCM_PLACR_DFCC_MASK)

#define MCM_PLACR_EFDS_MASK         0x4000u
#define MCM_PLACR_EFDS_SHIFT        14
#define MCM_PLACR_EFDS_WIDTH        1

```

```

#define MCM_PLACR_EFDS(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_EFDS_SHIFT))&MCM_PLACR_EFDS_MASK)

#define MCM_PLACR_DFCS_MASK          0x8000u
#define MCM_PLACR_DFCS_SHIFT        15
#define MCM_PLACR_DFCS_WIDTH        1

#define MCM_PLACR_DFCS(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCS_SHIFT))&MCM_PLACR_DFCS_MASK)

#define MCM_PLACR_ESFC_MASK          0x10000u
#define MCM_PLACR_ESFC_SHIFT        16
#define MCM_PLACR_ESFC_WIDTH        1

#define MCM_PLACR_ESFC(x)
(((uint32_t)((uint32_t)(x)<<MCM_PLACR_ESFC_SHIFT))&MCM_PLACR_ESFC_MASK)

/* CPO Bit Fields */

#define MCM_CPO_CPOREQ_MASK          0x1u
#define MCM_CPO_CPOREQ_SHIFT        0
#define MCM_CPO_CPOREQ_WIDTH        1

#define MCM_CPO_CPOREQ(x)
(((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOREQ_SHIFT))&MCM_CPO_CPOREQ_MASK)

#define MCM_CPO_CPOACK_MASK          0x2u
#define MCM_CPO_CPOACK_SHIFT        1
#define MCM_CPO_CPOACK_WIDTH        1

#define MCM_CPO_CPOACK(x)
(((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOACK_SHIFT))&MCM_CPO_CPOACK_MASK)

#define MCM_CPO_CPOWOI_MASK          0x4u
#define MCM_CPO_CPOWOI_SHIFT        2
#define MCM_CPO_CPOWOI_WIDTH        1

#define MCM_CPO_CPOWOI(x)
(((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOWOI_SHIFT))&MCM_CPO_CPOWOI_MASK)

/*
 * @}

```

```
/* /* end of group MCM_Register_Masks */
```

```
/* MCM - Peripheral instance base addresses */

/** Peripheral MCM base address */

#define MCM_BASE          (0xF0003000u)

/** Peripheral MCM base pointer */

#define MCM                ((MCM_Type *)MCM_BASE)

#define MCM_BASE_PTR        (MCM)

/** Array initializer of MCM peripheral base addresses */

#define MCM_BASE_ADDRS     { MCM_BASE }

/** Array initializer of MCM peripheral base pointers */

#define MCM_BASE_PTRS       { MCM }
```

```
/*
-----  
-- MCM - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros  
 * @{  
 */
```

```
/* MCM - Register instance definitions */

/* MCM */

#define MCM_PLASC          MCM_PLASC_REG(MCM)  
#define MCM_PLAMC         MCM_PLAMC_REG(MCM)  
#define MCM_PLACR         MCM_PLACR_REG(MCM)
```

```

#define MCM_CPO          MCM_CPO_REG(MCM)

/*
 * @}
 */

/* /* end of group MCM_Register_Accessor_Macros */

/*
 * @}
 */

/* /* end of group MCM_Peripheral_Access_Layer */

/*
-----*
-- MTB Peripheral Access Layer
----- */
/*!

* @addtogroup MTB_Peripheral_Access_Layer MTB Peripheral Access Layer
* @{
*/
/* ** MTB - Register Layout Typedef */

typedef struct {

    __IO uint32_t POSITION;           /**< MTB Position Register, offset: 0x0 */
    __IO uint32_t MASTER;            /**< MTB Master Register, offset: 0x4 */
    __IO uint32_t FLOW;              /**< MTB Flow Register, offset: 0x8 */
    __I  uint32_t BASE;              /**< MTB Base Register, offset: 0xC */
    uint8_t RESERVED_0[3824];
    __I  uint32_t MODECTRL;         /**< Integration Mode Control Register, offset: 0xF00 */
}

```

```

    uint8_t RESERVED_1[156];

    __I uint32_t TAGSET;           /**< Claim TAG Set Register, offset: 0xFA0 */
    __I uint32_t TAGCLEAR;        /**< Claim TAG Clear Register, offset: 0xFA4 */

    uint8_t RESERVED_2[8];

    __I uint32_t LOCKACCESS;      /**< Lock Access Register, offset: 0xFB0 */
    __I uint32_t LOCKSTAT;        /**< Lock Status Register, offset: 0xFB4 */
    __I uint32_t AUTHSTAT;        /**< Authentication Status Register, offset: 0xFB8 */
    __I uint32_t DEVICEARCH;      /**< Device Architecture Register, offset: 0xFBC */

    uint8_t RESERVED_3[8];

    __I uint32_t DEVICECFG;       /**< Device Configuration Register, offset: 0xFC8 */
    __I uint32_t DEVICETYPID;     /**< Device Type Identifier Register, offset: 0xFCC */
    __I uint32_t PERIPHID[8];     /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4
*/
    __I uint32_t COMPID[4];       /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
}

} MTB_Type, *MTB_MemMapPtr;

```

```

/*
-----  

-- MTB - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros  

* @{
* /
```

```

/* MTB - Register accessors */  

#define MTB_POSITION_REG(base) ((base)->POSITION)
```

```

#define MTB_MASTER_REG(base)           ((base)->MASTER)
#define MTB_FLOW_REG(base)             ((base)->FLOW)
#define MTB_BASE_REG(base)             ((base)->BASE)
#define MTB_MODECTRL_REG(base)         ((base)->MODECTRL)
#define MTB_TAGSET_REG(base)           ((base)->TAGSET)
#define MTB_TAGCLEAR_REG(base)         ((base)->TAGCLEAR)
#define MTB_LOCKACCESS_REG(base)       ((base)->LOCKACCESS)
#define MTB_LOCKSTAT_REG(base)         ((base)->LOCKSTAT)
#define MTB_AUTHSTAT_REG(base)         ((base)->AUTHSTAT)
#define MTB_DEVICEARCH_REG(base)       ((base)->DEVICEARCH)
#define MTB_DEVICECFG_REG(base)        ((base)->DEVICECFG)
#define MTB_DEVICETYPID_REG(base)      ((base)->DEVICETYPID)
#define MTB_PERIPHID_REG(base,index)   ((base)->PERIPHID[index])
#define MTB_PERIPHID_COUNT            8
#define MTB_COMPID_REG(base,index)     ((base)->COMPID[index])
#define MTB_COMPID_COUNT              4

```

```

/*!
* @}
*/
/* end of group MTB_Register_Accessor_Macros */

```

```

/*
-----*
-- MTB Register Masks
----- */

```

```

/*
* @addtogroup MTB_Register_Masks MTB Register Masks
* @{

```

*/

```
/* POSITION Bit Fields */

#define MTB_POSITION_WRAP_MASK          0x4u
#define MTB_POSITION_WRAP_SHIFT         2
#define MTB_POSITION_WRAP_WIDTH         1

#define MTB_POSITION_WRAP(x)
(((uint32_t)((uint32_t)(x))<<MTB_POSITION_WRAP_SHIFT))&MTB_POSITION_WRAP_MASK)

#define MTB_POSITION_POINTER_MASK        0xFFFFFFFF8u
#define MTB_POSITION_POINTER_SHIFT       3
#define MTB_POSITION_POINTER_WIDTH      29

#define MTB_POSITION_POINTER(x)
(((uint32_t)((uint32_t)(x))<<MTB_POSITION_POINTER_SHIFT))&MTB_POSITION_POINTER_MASK)

/* MASTER Bit Fields */

#define MTB_MASTER_MASK_MASK           0x1Fu
#define MTB_MASTER_MASK_SHIFT          0
#define MTB_MASTER_MASK_WIDTH          5

#define MTB_MASTER_MASK(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_MASK_SHIFT))&MTB_MASTER_MASK_MASK)

#define MTB_MASTER_TSTARTEN_MASK        0x20u
#define MTB_MASTER_TSTARTEN_SHIFT       5
#define MTB_MASTER_TSTARTEN_WIDTH       1

#define MTB_MASTER_TSTARTEN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTARTEN_SHIFT))&MTB_MASTER_TSTARTEN_MASK)

#define MTB_MASTER_TSTOPEN_MASK         0x40u
#define MTB_MASTER_TSTOPEN_SHIFT        6
#define MTB_MASTER_TSTOPEN_WIDTH        1

#define MTB_MASTER_TSTOPEN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTOPEN_SHIFT))&MTB_MASTER_TSTOPEN_MASK)

#define MTB_MASTER_SFRWPRI_MASk        0x80u
```

```

#define MTB_MASTER_SFRWPRI_V_SHIFT      7
#define MTB_MASTER_SFRWPRI_V_WIDTH      1
#define MTB_MASTER_SFRWPRI(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_SFRWPRI_SHIFT))&MTB_MASTER_SFRWPRI_MASK)

#define MTB_MASTER_RAMPRIV_MASK        0x100u
#define MTB_MASTER_RAMPRIV_SHIFT       8
#define MTB_MASTER_RAMPRIV_WIDTH       1
#define MTB_MASTER_RAMPRIV(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_RAMPRIV_SHIFT))&MTB_MASTER_RAMPRIV_MASK)

#define MTB_MASTER_HALTREQ_MASK        0x200u
#define MTB_MASTER_HALTREQ_SHIFT       9
#define MTB_MASTER_HALTREQ_WIDTH       1
#define MTB_MASTER_HALTREQ(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_HALTREQ_SHIFT))&MTB_MASTER_HALTREQ_MASK)

#define MTB_MASTER_EN_MASK            0x80000000u
#define MTB_MASTER_EN_SHIFT           31
#define MTB_MASTER_EN_WIDTH           1
#define MTB_MASTER_EN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_EN_SHIFT))&MTB_MASTER_EN_MASK)

/* FLOW Bit Fields */

#define MTB_FLOW_AUTOSTOP_MASK         0x1u
#define MTB_FLOW_AUTOSTOP_SHIFT        0
#define MTB_FLOW_AUTOSTOP_WIDTH        1
#define MTB_FLOW_AUTOSTOP(x)
(((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOSTOP_SHIFT))&MTB_FLOW_AUTOSTOP_MASK)

#define MTB_FLOW_AUTOHALT_MASK         0x2u
#define MTB_FLOW_AUTOHALT_SHIFT        1
#define MTB_FLOW_AUTOHALT_WIDTH        1
#define MTB_FLOW_AUTOHALT(x)
(((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOHALT_SHIFT))&MTB_FLOW_AUTOHALT_MASK)

#define MTB_FLOW_WATERMARK_MASK        0xFFFFFFF8u

```

```

#define MTB_FLOW_WATERMARK_SHIFT      3
#define MTB_FLOW_WATERMARK_WIDTH     29
#define MTB_FLOW_WATERMARK(x)
(((uint32_t)((uint32_t)(x))<<MTB_FLOW_WATERMARK_SHIFT))&MTB_FLOW_WATERMARK_MASK)

/* BASE Bit Fields */

#define MTB_BASE_BASEADDR_MASK        0xFFFFFFFFu
#define MTB_BASE_BASEADDR_SHIFT       0
#define MTB_BASE_BASEADDR_WIDTH      32
#define MTB_BASE_BASEADDR(x)
(((uint32_t)((uint32_t)(x))<<MTB_BASE_BASEADDR_SHIFT))&MTB_BASE_BASEADDR_MASK

/* MODECTRL Bit Fields */

#define MTB_MODECTRL_MODECTRL_MASK    0xFFFFFFFFu
#define MTB_MODECTRL_MODECTRL_SHIFT   0
#define MTB_MODECTRL_MODECTRL_WIDTH  32
#define MTB_MODECTRL_MODECTRL(x)
(((uint32_t)((uint32_t)(x))<<MTB_MODECTRL_MODECTRL_SHIFT))&MTB_MODECTRL_MODECTRL_MASK

/* TAGSET Bit Fields */

#define MTB_TAGSET_TAGSET_MASK        0xFFFFFFFFu
#define MTB_TAGSET_TAGSET_SHIFT       0
#define MTB_TAGSET_TAGSET_WIDTH      32
#define MTB_TAGSET_TAGSET(x)
(((uint32_t)((uint32_t)(x))<<MTB_TAGSET_TAGSET_SHIFT))&MTB_TAGSET_TAGSET_MASK

/* TAGCLEAR Bit Fields */

#define MTB_TAGCLEAR_TAGCLEAR_MASK    0xFFFFFFFFu
#define MTB_TAGCLEAR_TAGCLEAR_SHIFT   0
#define MTB_TAGCLEAR_TAGCLEAR_WIDTH  32
#define MTB_TAGCLEAR_TAGCLEAR(x)
(((uint32_t)((uint32_t)(x))<<MTB_TAGCLEAR_TAGCLEAR_SHIFT))&MTB_TAGCLEAR_TAGCLEAR_MASK

/* LOCKACCESS Bit Fields */

#define MTB_LOCKACCESS_LOCKACCESS_MASK 0xFFFFFFFFu

```

```

#define MTB_LOCKACCESS_LOCKACCESS_SHIFT      0
#define MTB_LOCKACCESS_LOCKACCESS_WIDTH      32
#define MTB_LOCKACCESS_LOCKACCESS(x)
(((uint32_t)((uint32_t)(x)<<MTB_LOCKACCESS_LOCKACCESS_SHIFT))&MTB_LOCKACCESS_LOCKACCESS_MASK)

/* LOCKSTAT Bit Fields */

#define MTB_LOCKSTAT_LOCKSTAT_MASK          0xFFFFFFFFu
#define MTB_LOCKSTAT_LOCKSTAT_SHIFT         0
#define MTB_LOCKSTAT_LOCKSTAT_WIDTH         32
#define MTB_LOCKSTAT_LOCKSTAT(x)
(((uint32_t)((uint32_t)(x)<<MTB_LOCKSTAT_LOCKSTAT_SHIFT))&MTB_LOCKSTAT_LOCKSTAT_MASK)

/* AUTHSTAT Bit Fields */

#define MTB_AUTHSTAT_BIT0_MASK              0x1u
#define MTB_AUTHSTAT_BIT0_SHIFT             0
#define MTB_AUTHSTAT_BIT0_WIDTH             1
#define MTB_AUTHSTAT_BIT0(x)
(((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT0_SHIFT))&MTB_AUTHSTAT_BIT0_MASK)

#define MTB_AUTHSTAT_BIT1_MASK              0x2u
#define MTB_AUTHSTAT_BIT1_SHIFT             1
#define MTB_AUTHSTAT_BIT1_WIDTH             1
#define MTB_AUTHSTAT_BIT1(x)
(((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT1_SHIFT))&MTB_AUTHSTAT_BIT1_MASK)

#define MTB_AUTHSTAT_BIT2_MASK              0x4u
#define MTB_AUTHSTAT_BIT2_SHIFT             2
#define MTB_AUTHSTAT_BIT2_WIDTH             1
#define MTB_AUTHSTAT_BIT2(x)
(((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT2_SHIFT))&MTB_AUTHSTAT_BIT2_MASK)

#define MTB_AUTHSTAT_BIT3_MASK              0x8u
#define MTB_AUTHSTAT_BIT3_SHIFT             3
#define MTB_AUTHSTAT_BIT3_WIDTH             1

```

```

#define MTB_AUTHSTAT_BIT3(x)
(((uint32_t)((uint32_t)(x))<<MTB_AUTHSTAT_BIT3_SHIFT))&MTB_AUTHSTAT_BIT3_MASK)

/* DEVICEARCH Bit Fields */

#define MTB_DEVICEARCH_DEVICEARCH_MASK      0xFFFFFFFFu
#define MTB_DEVICEARCH_DEVICEARCH_SHIFT     0
#define MTB_DEVICEARCH_DEVICEARCH_WIDTH     32

#define MTB_DEVICEARCH_DEVICEARCH(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICEARCH_DEVICEARCH_SHIFT))&MTB_DEVICEARCH_DEVICEARCH_MASK

/* DEVICECFG Bit Fields */

#define MTB_DEVICECFG_DEVICECFG_MASK      0xFFFFFFFFu
#define MTB_DEVICECFG_DEVICECFG_SHIFT     0
#define MTB_DEVICECFG_DEVICECFG_WIDTH     32

#define MTB_DEVICECFG_DEVICECFG(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICECFG_DEVICECFG_SHIFT))&MTB_DEVICECFG_DEVICECFG_MASK

/* DEVICETYPID Bit Fields */

#define MTB_DEVICETYPID_DEVICETYPID_MASK   0xFFFFFFFFu
#define MTB_DEVICETYPID_DEVICETYPID_SHIFT  0
#define MTB_DEVICETYPID_DEVICETYPID_WIDTH  32

#define MTB_DEVICETYPID_DEVICETYPID(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICETYPID_DEVICETYPID_SHIFT))&MTB_DEVICETYPID_DEVICETYPID_MASK

/* PERIPHID Bit Fields */

#define MTB_PERIPHID_PERIPHID_MASK      0xFFFFFFFFu
#define MTB_PERIPHID_PERIPHID_SHIFT     0
#define MTB_PERIPHID_PERIPHID_WIDTH    32

#define MTB_PERIPHID_PERIPHID(x)
(((uint32_t)((uint32_t)(x))<<MTB_PERIPHID_PERIPHID_SHIFT))&MTB_PERIPHID_PERIPHID_MASK

/* COMPID Bit Fields */

#define MTB_COMPID_COMPID_MASK      0xFFFFFFFFu
#define MTB_COMPID_COMPID_SHIFT     0

```

```

#define MTB_COMPID_COMPID_WIDTH          32

#define MTB_COMPID_COMPID(x)
((uint32_t)((uint32_t)(x)<<MTB_COMPID_COMPID_SHIFT))&MTB_COMPID_COMPID_MASK)

/*
 * @}
 */

/* /* end of group MTB_Register_Masks */

/* MTB - Peripheral instance base addresses */
/** Peripheral MTB base address */
#define MTB_BASE              (0xF0000000u)

/** Peripheral MTB base pointer */
#define MTB           ((MTB_Type *)MTB_BASE)

#define MTB_BASE_PTR          (MTB)

/** Array initializer of MTB peripheral base addresses */
#define MTB_BASE_ADDRS        { MTB_BASE }

/** Array initializer of MTB peripheral base pointers */
#define MTB_BASE_PTRS         { MTB }

/*
----- --
-- MTB - Register accessor macros
----- */

```

```

/*
 * @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros
 * @{
 */


```

```
/* MTB - Register instance definitions */

/* MTB */

#define MTB_POSITION          MTB_POSITION_REG(MTB)
#define MTB_MASTER             MTB_MASTER_REG(MTB)
#define MTB_FLOW               MTB_FLOW_REG(MTB)
#define MTB_BASEr              MTB_BASE_REG(MTB)
#define MTB_MODECTRL           MTB_MODECTRL_REG(MTB)
#define MTB_TAGSET              MTB_TAGSET_REG(MTB)
#define MTB_TAGCLEAR            MTB_TAGCLEAR_REG(MTB)
#define MTB_LOCKACCESS           MTB_LOCKACCESS_REG(MTB)
#define MTB_LOCKSTAT             MTB_LOCKSTAT_REG(MTB)
#define MTB_AUTHSTAT             MTB_AUTHSTAT_REG(MTB)
#define MTB_DEVICESEARCH          MTB_DEVICESEARCH_REG(MTB)
#define MTB_DEVICECFG             MTB_DEVICECFG_REG(MTB)
#define MTB_DEVICETYPID          MTB_DEVICETYPID_REG(MTB)
#define MTB_PERIPHID4            MTB_PERIPHID_REG(MTB,0)
#define MTB_PERIPHID5            MTB_PERIPHID_REG(MTB,1)
#define MTB_PERIPHID6            MTB_PERIPHID_REG(MTB,2)
#define MTB_PERIPHID7            MTB_PERIPHID_REG(MTB,3)
#define MTB_PERIPHID0             MTB_PERIPHID_REG(MTB,4)
#define MTB_PERIPHID1             MTB_PERIPHID_REG(MTB,5)
#define MTB_PERIPHID2             MTB_PERIPHID_REG(MTB,6)
#define MTB_PERIPHID3             MTB_PERIPHID_REG(MTB,7)
#define MTB_COMPID0               MTB_COMPID_REG(MTB,0)
#define MTB_COMPID1               MTB_COMPID_REG(MTB,1)
#define MTB_COMPID2               MTB_COMPID_REG(MTB,2)
#define MTB_COMPID3               MTB_COMPID_REG(MTB,3)
```

```
/* MTB - Register array accessors */

#define MTB_PERIPHID(index)          MTB_PERIPHID_REG(MTB,index)
#define MTB_COMPID(index)            MTB_COMPID_REG(MTB,index)

/*!
 * @}

 */ /* end of group MTB_Register_Accessor_Macros */
```

```
/*!
 * @}

 */ /* end of group MTB_Peripheral_Access_Layer */
```

```
/*
-----  
-- MTBDWT Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup MTBDWT_Peripheral_Access_Layer MTBDWT Peripheral Access Layer  
 * @{  
 */
```

```
/** MTBDWT - Register Layout Typedef */  
  
typedef struct {  
    __I uint32_t CTRL;           /**< MTB DWT Control Register, offset: 0x0 */  
    uint8_t RESERVED_0[28];  
    struct {                  /* offset: 0x20, array step: 0x10 */  
        /*  
         *  
         */  
    };  
};
```

```

    __IO uint32_t COMP;           /**< MTB_DWT Comparator Register, array offset: 0x20, array
step: 0x10 */

    __IO uint32_t MASK;          /**< MTB_DWT Comparator Mask Register, array offset: 0x24,
array step: 0x10 */

    __IO uint32_t FCT;           /**< MTB_DWT Comparator Function Register 0..MTB_DWT
Comparator Function Register 1, array offset: 0x28, array step: 0x10 */

    uint8_t RESERVED_0[4];

} COMPARATOR[2];

    uint8_t RESERVED_1[448];

__IO uint32_t TBCTRL;           /**< MTB_DWT Trace Buffer Control Register, offset: 0x200 */

    uint8_t RESERVED_2[3524];

__I uint32_t DEVICECFG;         /**< Device Configuration Register, offset: 0xFC8 */

__I uint32_t DEVICETYPID;       /**< Device Type Identifier Register, offset: 0xFCC */

__I uint32_t PERIPHID[8];       /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4
*/
    __I uint32_t COMPID[4];        /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
}

} MTBDWT_Type, *MTBDWT_MemMapPtr;

```

```

/*
-----*
-- MTBDWT - Register accessor macros
----- */

```

```

/*!
* @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros
* @{
*/

```

```

/* MTBDWT - Register accessors */

```

```

#define MTBDWT_CTRL_REG(base)           ((base)->CTRL)
#define MTBDWT_COMP_REG(base,index)     ((base)->COMPARATOR[index].COMP)
#define MTBDWT_COMP_COUNT             2
#define MTBDWT_MASK_REG(base,index)    ((base)->COMPARATOR[index].MASK)
#define MTBDWT_MASK_COUNT             2
#define MTBDWT_FCT_REG(base,index)     ((base)->COMPARATOR[index].FCT)
#define MTBDWT_FCT_COUNT              2
#define MTBDWT_TBCTRL_REG(base)        ((base)->TBCTRL)
#define MTBDWT_DEVICECFG_REG(base)     ((base)->DEVICECFG)
#define MTBDWT_DEVICETYPID_REG(base)   ((base)->DEVICETYPID)
#define MTBDWT_PERIPHID_REG(base,index) ((base)->PERIPHID[index])
#define MTBDWT_PERIPHID_COUNT         8
#define MTBDWT_COMPID_REG(base,index)  ((base)->COMPID[index])
#define MTBDWT_COMPID_COUNT            4

/*!
 * @}
 */

/* /* end of group MTBDWT_Register_Accessor_Macros */

```

```

/*
-----  

-- MTBDWT Register Masks  

----- */

```

```

/*!  

* @addtogroup MTBDWT_Register_Masks MTBDWT Register Masks  

* @{
* /
```

```

/* CTRL Bit Fields */

#define MTBDWT_CTRL_DWTCFGCTRL_MASK      0xFFFFFFFFu
#define MTBDWT_CTRL_DWTCFGCTRL_SHIFT     0
#define MTBDWT_CTRL_DWTCFGCTRL_WIDTH     28
#define MTBDWT_CTRL_DWTCFGCTRL(x)        (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_DWTCFGCTRL_SHIFT))&MTBDWT_CTRL_DWTCFGCTRL_MASK)

#define MTBDWT_CTRL_NUMCMP_MASK          0xF0000000u
#define MTBDWT_CTRL_NUMCMP_SHIFT         28
#define MTBDWT_CTRL_NUMCMP_WIDTH         4
#define MTBDWT_CTRL_NUMCMP(x)           (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_NUMCMP_SHIFT))&MTBDWT_CTRL_NUMCMP_MASK)

/* COMP Bit Fields */

#define MTBDWT_COMP_COMP_MASK           0xFFFFFFFFu
#define MTBDWT_COMP_COMP_SHIFT          0
#define MTBDWT_COMP_COMP_WIDTH          32
#define MTBDWT_COMP_COMP(x)             (((uint32_t)((uint32_t)(x))<<MTBDWT_COMP_COMP_SHIFT))&MTBDWT_COMP_COMP_MASK)

/* MASK Bit Fields */

#define MTBDWT_MASK_MASK_MASK          0x1Fu
#define MTBDWT_MASK_MASK_SHIFT          0
#define MTBDWT_MASK_MASK_WIDTH          5
#define MTBDWT_MASK_MASK(x)             (((uint32_t)((uint32_t)(x))<<MTBDWT_MASK_MASK_SHIFT))&MTBDWT_MASK_MASK_MASK)

/* FCT Bit Fields */

#define MTBDWT_FCT_FUNCTION_MASK        0xFu
#define MTBDWT_FCT_FUNCTION_SHIFT       0
#define MTBDWT_FCT_FUNCTION_WIDTH       4
#define MTBDWT_FCT_FUNCTION(x)          (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_FUNCTION_SHIFT))&MTBDWT_FCT_FUNCTION_MASK)

#define MTBDWT_FCT_DATAVMATCH_MASK      0x100u

```

```

#define MTBDWT_FCT_DATAVMATCH_SHIFT      8
#define MTBDWT_FCT_DATAVMATCH_WIDTH      1
#define MTBDWT_FCT_DATAVMATCH(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVMATCH_SHIFT))&MTBDWT_FCT_DATAVMATCH_MASK

#define MTBDWT_FCT_DATAVSIZE_MASK        0xC00u
#define MTBDWT_FCT_DATAVSIZE_SHIFT       10
#define MTBDWT_FCT_DATAVSIZE_WIDTH       2
#define MTBDWT_FCT_DATAVSIZE(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVSIZE_SHIFT))&MTBDWT_FCT_DATAVSIZE_MASK

#define MTBDWT_FCT_DATAVADDR0_MASK       0xF000u
#define MTBDWT_FCT_DATAVADDR0_SHIFT      12
#define MTBDWT_FCT_DATAVADDR0_WIDTH      4
#define MTBDWT_FCT_DATAVADDR0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVADDR0_SHIFT))&MTBDWT_FCT_DATAVADDR0_MASK

#define MTBDWT_FCT_MATCHED_MASK         0x1000000u
#define MTBDWT_FCT_MATCHED_SHIFT        24
#define MTBDWT_FCT_MATCHED_WIDTH        1
#define MTBDWT_FCT_MATCHED(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_MATCHED_SHIFT))&MTBDWT_FCT_MATCHED_MASK

/* TBCTRL Bit Fields */

#define MTBDWT_TBCTRL_ACOMP0_MASK        0x1u
#define MTBDWT_TBCTRL_ACOMP0_SHIFT       0
#define MTBDWT_TBCTRL_ACOMP0_WIDTH       1
#define MTBDWT_TBCTRL_ACOMP0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP0_SHIFT))&MTBDWT_TBCTRL_ACOMP0_MASK

#define MTBDWT_TBCTRL_ACOMP1_MASK        0x2u
#define MTBDWT_TBCTRL_ACOMP1_SHIFT       1
#define MTBDWT_TBCTRL_ACOMP1_WIDTH       1
#define MTBDWT_TBCTRL_ACOMP1(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP1_SHIFT))&MTBDWT_TBCTRL_ACOMP1_MASK

```

```

#define MTBDWT_TBCTRL_NUMCOMP_MASK      0xF0000000u
#define MTBDWT_TBCTRL_NUMCOMP_SHIFT     28
#define MTBDWT_TBCTRL_NUMCOMP_WIDTH     4
#define MTBDWT_TBCTRL_NUMCOMP(x)        (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_NUMCOMP_SHIFT))&MTBDWT_TBCTRL_NUMCOMP_M
ASK)

/* DEVICECFG Bit Fields */

#define MTBDWT_DEVICECFG_DEVICECFG_MASK    0xFFFFFFFFu
#define MTBDWT_DEVICECFG_DEVICECFG_SHIFT   0
#define MTBDWT_DEVICECFG_DEVICECFG_WIDTH   32
#define MTBDWT_DEVICECFG_DEVICECFG(x)       (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICECFG_DEVICECFG_SHIFT))&MTBDWT_DEVICECFG_DEVICE
CFG_MASK)

/* DEVICETYPID Bit Fields */

#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK 0xFFFFFFFFu
#define MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT 0
#define MTBDWT_DEVICETYPID_DEVICETYPID_WIDTH 32
#define MTBDWT_DEVICETYPID_DEVICETYPID(x)   (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT))&MTBDWT_DEVICETYPID_DE
VICETYPID_MASK)

/* PERIPHID Bit Fields */

#define MTBDWT_PERIPHID_PERIPHID_MASK     0xFFFFFFFFu
#define MTBDWT_PERIPHID_PERIPHID_SHIFT    0
#define MTBDWT_PERIPHID_PERIPHID_WIDTH    32
#define MTBDWT_PERIPHID_PERIPHID(x)       (((uint32_t)((uint32_t)(x))<<MTBDWT_PERIPHID_PERIPHID_SHIFT))&MTBDWT_PERIPHID_PERIPHID_M
ASK)

/* COMPID Bit Fields */

#define MTBDWT_COMPID_COMPID_MASK         0xFFFFFFFFu
#define MTBDWT_COMPID_COMPID_SHIFT        0
#define MTBDWT_COMPID_COMPID_WIDTH        32

```

```

#define MTBDWT_COMPID_COMPID(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_COMPID_COMPID_SHIFT))&MTBDWT_COMPID_COMPID_MASK

/*
 * @}
 */

/* /* end of group MTBDWT_Register_Masks */

/* MTBDWT - Peripheral instance base addresses */
/** Peripheral MTBDWT base address */
#define MTBDWT_BASE          (0xF0001000u)
/** Peripheral MTBDWT base pointer */
#define MTBDWT          ((MTBDWT_Type *)MTBDWT_BASE)
#define MTBDWT_BASE_PTR      (MTBDWT)
/** Array initializer of MTBDWT peripheral base addresses */
#define MTBDWT_BASE_ADDRS     { MTBDWT_BASE }
/** Array initializer of MTBDWT peripheral base pointers */
#define MTBDWT_BASE_PTRS      { MTBDWT }

/*
-----  

-- MTBDWT - Register accessor macros  

----- */
/*!  

* @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros  

* @{
*/

```

```

/* MTBDWT - Register instance definitions */

/* MTBDWT */

#define MTBDWT_CTRL           MTBDWT_CTRL_REG(MTBDWT)
#define MTBDWT_COMP0          MTBDWT_COMP_REG(MTBDWT,0)
#define MTBDWT_MASK0          MTBDWT_MASK_REG(MTBDWT,0)
#define MTBDWT_FCT0           MTBDWT_FCT_REG(MTBDWT,0)
#define MTBDWT_COMP1          MTBDWT_COMP_REG(MTBDWT,1)
#define MTBDWT_MASK1          MTBDWT_MASK_REG(MTBDWT,1)
#define MTBDWT_FCT1           MTBDWT_FCT_REG(MTBDWT,1)
#define MTBDWT_TBCTRL         MTBDWT_TBCTRL_REG(MTBDWT)
#define MTBDWT_DEVICECFG      MTBDWT_DEVICECFG_REG(MTBDWT)
#define MTBDWT_DEVICETYPID    MTBDWT_DEVICETYPID_REG(MTBDWT)
#define MTBDWT_PERIPHID4      MTBDWT_PERIPHID_REG(MTBDWT,0)
#define MTBDWT_PERIPHID5      MTBDWT_PERIPHID_REG(MTBDWT,1)
#define MTBDWT_PERIPHID6      MTBDWT_PERIPHID_REG(MTBDWT,2)
#define MTBDWT_PERIPHID7      MTBDWT_PERIPHID_REG(MTBDWT,3)
#define MTBDWT_PERIPHID0      MTBDWT_PERIPHID_REG(MTBDWT,4)
#define MTBDWT_PERIPHID1      MTBDWT_PERIPHID_REG(MTBDWT,5)
#define MTBDWT_PERIPHID2      MTBDWT_PERIPHID_REG(MTBDWT,6)
#define MTBDWT_PERIPHID3      MTBDWT_PERIPHID_REG(MTBDWT,7)
#define MTBDWT_COMPID0         MTBDWT_COMPID_REG(MTBDWT,0)
#define MTBDWT_COMPID1         MTBDWT_COMPID_REG(MTBDWT,1)
#define MTBDWT_COMPID2         MTBDWT_COMPID_REG(MTBDWT,2)
#define MTBDWT_COMPID3         MTBDWT_COMPID_REG(MTBDWT,3)

/* MTBDWT - Register array accessors */

#define MTBDWT_COMP(index)     MTBDWT_COMP_REG(MTBDWT,index)
#define MTBDWT_MASK(index)     MTBDWT_MASK_REG(MTBDWT,index)
#define MTBDWT_FCT(index)      MTBDWT_FCT_REG(MTBDWT,index)

```

```
#define MTBDWT_PERIPHID(index)          MTBDWT_PERIPHID_REG(MTBDWT,index)
#define MTBDWT_COMPID(index)              MTBDWT_COMPID_REG(MTBDWT,index)
```

```
/*!
 * @}
 */
/* end of group MTBDWT_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group MTBDWT_Peripheral_Access_Layer */
```

```
/*
-----  
-- NV Peripheral Access Layer  
----- */
```

```
/*!  
* @addtogroup NV_Peripheral_Access_Layer NV Peripheral Access Layer  
* @{  
*/
```

```
/** NV - Register Layout Typedef */
```

```
typedef struct {  
    __I uint8_t BACKKEY3;           /**< Backdoor Comparison Key 3., offset: 0x0 */  
    __I uint8_t BACKKEY2;           /**< Backdoor Comparison Key 2., offset: 0x1 */  
    __I uint8_t BACKKEY1;           /**< Backdoor Comparison Key 1., offset: 0x2 */  
    __I uint8_t BACKKEY0;           /**< Backdoor Comparison Key 0., offset: 0x3 */  
    __I uint8_t BACKKEY7;           /**< Backdoor Comparison Key 7., offset: 0x4 */
```

```

__I uint8_t BACKKEY6;           /**< Backdoor Comparison Key 6., offset: 0x5 */
__I uint8_t BACKKEY5;           /**< Backdoor Comparison Key 5., offset: 0x6 */
__I uint8_t BACKKEY4;           /**< Backdoor Comparison Key 4., offset: 0x7 */
__I uint8_t FPROT3;            /**< Non-volatile P-Flash Protection 1 - Low Register, offset: 0x8
*/
__I uint8_t FPROT2;            /**< Non-volatile P-Flash Protection 1 - High Register, offset: 0x9
*/
__I uint8_t FPROT1;            /**< Non-volatile P-Flash Protection 0 - Low Register, offset: 0xA
*/
__I uint8_t FPROT0;            /**< Non-volatile P-Flash Protection 0 - High Register, offset: 0xB
*/
__I uint8_t FSEC;              /**< Non-volatile Flash Security Register, offset: 0xC */
__I uint8_t FOPT;              /**< Non-volatile Flash Option Register, offset: 0xD */

} NV_Type, *NV_MemMapPtr;

```

```

/*
-----*
-- NV - Register accessor macros
----- */

```

```

/*!

```

```

* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros
* @{
*/

```

```

/* NV - Register accessors */

```

```

#define NV_BACKKEY3_REG(base)      ((base)->BACKKEY3)
#define NV_BACKKEY2_REG(base)      ((base)->BACKKEY2)
#define NV_BACKKEY1_REG(base)      ((base)->BACKKEY1)
#define NV_BACKKEY0_REG(base)      ((base)->BACKKEY0)

```

```
#define NV_BACKKEY7_REG(base)          ((base)->BACKKEY7)
#define NV_BACKKEY6_REG(base)          ((base)->BACKKEY6)
#define NV_BACKKEY5_REG(base)          ((base)->BACKKEY5)
#define NV_BACKKEY4_REG(base)          ((base)->BACKKEY4)
#define NV_FPROT3_REG(base)           ((base)->FPROT3)
#define NV_FPROT2_REG(base)           ((base)->FPROT2)
#define NV_FPROT1_REG(base)           ((base)->FPROT1)
#define NV_FPROT0_REG(base)           ((base)->FPROT0)
#define NV_FSEC_REG(base)             ((base)->FSEC)
#define NV_FOPT_REG(base)             ((base)->FOPT)
```

```
/*
 * @}
 */
/* /* end of group NV_Register_Accessor_Macros */
```

```
/*
-----  
-- NV Register Masks  
----- */
```

```
/*
 * @addtogroup NV_Register_Masks NV Register Masks
 * @{
 */

```

```
/* BACKKEY3 Bit Fields */

#define NV_BACKKEY3_KEY_MASK          0xFFu
#define NV_BACKKEY3_KEY_SHIFT         0
#define NV_BACKKEY3_KEY_WIDTH         8
```

```

#define NV_BACKKEY3_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY3_KEY_SHIFT))&NV_BACKKEY3_KEY_MASK)

/* BACKKEY2 Bit Fields */

#define NV_BACKKEY2_KEY_MASK          0xFFu
#define NV_BACKKEY2_KEY_SHIFT         0
#define NV_BACKKEY2_KEY_WIDTH         8

#define NV_BACKKEY2_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY2_KEY_SHIFT))&NV_BACKKEY2_KEY_MASK)

/* BACKKEY1 Bit Fields */

#define NV_BACKKEY1_KEY_MASK          0xFFu
#define NV_BACKKEY1_KEY_SHIFT         0
#define NV_BACKKEY1_KEY_WIDTH         8

#define NV_BACKKEY1_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY1_KEY_SHIFT))&NV_BACKKEY1_KEY_MASK)

/* BACKKEY0 Bit Fields */

#define NV_BACKKEY0_KEY_MASK          0xFFu
#define NV_BACKKEY0_KEY_SHIFT         0
#define NV_BACKKEY0_KEY_WIDTH         8

#define NV_BACKKEY0_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY0_KEY_SHIFT))&NV_BACKKEY0_KEY_MASK)

/* BACKKEY7 Bit Fields */

#define NV_BACKKEY7_KEY_MASK          0xFFu
#define NV_BACKKEY7_KEY_SHIFT         0
#define NV_BACKKEY7_KEY_WIDTH         8

#define NV_BACKKEY7_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY7_KEY_SHIFT))&NV_BACKKEY7_KEY_MASK)

/* BACKKEY6 Bit Fields */

#define NV_BACKKEY6_KEY_MASK          0xFFu
#define NV_BACKKEY6_KEY_SHIFT         0
#define NV_BACKKEY6_KEY_WIDTH         8

```

```

#define NV_BACKKEY6_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY6_KEY_SHIFT))&NV_BACKKEY6_KEY_MASK)

/* BACKKEY5 Bit Fields */

#define NV_BACKKEY5_KEY_MASK          0xFFu
#define NV_BACKKEY5_KEY_SHIFT         0
#define NV_BACKKEY5_KEY_WIDTH         8

#define NV_BACKKEY5_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY5_KEY_SHIFT))&NV_BACKKEY5_KEY_MASK)

/* BACKKEY4 Bit Fields */

#define NV_BACKKEY4_KEY_MASK          0xFFu
#define NV_BACKKEY4_KEY_SHIFT         0
#define NV_BACKKEY4_KEY_WIDTH         8

#define NV_BACKKEY4_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY4_KEY_SHIFT))&NV_BACKKEY4_KEY_MASK)

/* FPROT3 Bit Fields */

#define NV_FPROT3_PROT_MASK          0xFFu
#define NV_FPROT3_PROT_SHIFT         0
#define NV_FPROT3_PROT_WIDTH         8

#define NV_FPROT3_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT3_PROT_SHIFT))&NV_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

#define NV_FPROT2_PROT_MASK          0xFFu
#define NV_FPROT2_PROT_SHIFT         0
#define NV_FPROT2_PROT_WIDTH         8

#define NV_FPROT2_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT2_PROT_SHIFT))&NV_FPROT2_PROT_MASK)

/* FPROT1 Bit Fields */

#define NV_FPROT1_PROT_MASK          0xFFu
#define NV_FPROT1_PROT_SHIFT         0
#define NV_FPROT1_PROT_WIDTH         8

```

```

#define NV_FPROT1_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT1_PROT_SHIFT))&NV_FPROT1_PROT_MASK)

/* FPROT0 Bit Fields */

#define NV_FPROT0_PROT_MASK          0xFFu
#define NV_FPROT0_PROT_SHIFT         0
#define NV_FPROT0_PROT_WIDTH         8

#define NV_FPROT0_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT0_PROT_SHIFT))&NV_FPROT0_PROT_MASK)

/* FSEC Bit Fields */

#define NV_FSEC_SEC_MASK            0x3u
#define NV_FSEC_SEC_SHIFT           0
#define NV_FSEC_SEC_WIDTH           2

#define NV_FSEC_SEC(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_SEC_SHIFT))&NV_FSEC_SEC_MASK)

#define NV_FSEC_FSLACC_MASK          0xCu
#define NV_FSEC_FSLACC_SHIFT         2
#define NV_FSEC_FSLACC_WIDTH         2

#define NV_FSEC_FSLACC(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_FSLACC_SHIFT))&NV_FSEC_FSLACC_MASK)

#define NV_FSEC_MEEN_MASK            0x30u
#define NV_FSEC_MEEN_SHIFT           4
#define NV_FSEC_MEEN_WIDTH           2

#define NV_FSEC_MEEN(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_MEEN_SHIFT))&NV_FSEC_MEEN_MASK)

#define NV_FSEC_KEYEN_MASK           0xC0u
#define NV_FSEC_KEYEN_SHIFT          6
#define NV_FSEC_KEYEN_WIDTH           2

#define NV_FSEC_KEYEN(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_KEYEN_SHIFT))&NV_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define NV_FOPT_LPBOOT0_MASK          0x1u

```

```

#define NV_FOPT_LPBOOT0_SHIFT          0
#define NV_FOPT_LPBOOT0_WIDTH          1
#define NV_FOPT_LPBOOT0(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT0_SHIFT))&NV_FOPT_LPBOOT0_MASK)

#define NV_FOPT_NMI_DIS_MASK          0x4u
#define NV_FOPT_NMI_DIS_SHIFT         2
#define NV_FOPT_NMI_DIS_WIDTH         1
#define NV_FOPT_NMI_DIS(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_NMI_DIS_SHIFT))&NV_FOPT_NMI_DIS_MASK)

#define NV_FOPT_RESET_PIN_CFG_MASK    0x8u
#define NV_FOPT_RESET_PIN_CFG_SHIFT   3
#define NV_FOPT_RESET_PIN_CFG_WIDTH   1
#define NV_FOPT_RESET_PIN_CFG(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_RESET_PIN_CFG_SHIFT))&NV_FOPT_RESET_PIN_CFG_MASK)

#define NV_FOPT_LPBOOT1_MASK          0x10u
#define NV_FOPT_LPBOOT1_SHIFT         4
#define NV_FOPT_LPBOOT1_WIDTH         1
#define NV_FOPT_LPBOOT1(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT1_SHIFT))&NV_FOPT_LPBOOT1_MASK)

#define NV_FOPT_FAST_INIT_MASK        0x20u
#define NV_FOPT_FAST_INIT_SHIFT       5
#define NV_FOPT_FAST_INIT_WIDTH       1
#define NV_FOPT_FAST_INIT(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_FAST_INIT_SHIFT))&NV_FOPT_FAST_INIT_MASK)

/*
 * @}
 */ /* end of group NV_Register_Masks */

```

```

/* NV - Peripheral instance base addresses */

/** Peripheral FTFA_FlashConfig base address */

#define FTFA_FlashConfig_BASE          (0x400u)

/** Peripheral FTFA_FlashConfig base pointer */

#define FTFA_FlashConfig             ((NV_Type *)FTFA_FlashConfig_BASE)

#define FTFA_FlashConfig_BASE_PTR    (FTFA_FlashConfig)

/** Array initializer of NV peripheral base addresses */

#define NV_BASE_ADDRS               { FTFA_FlashConfig_BASE }

/** Array initializer of NV peripheral base pointers */

#define NV_BASE_PTRS                { FTFA_FlashConfig }

```

```

/*
-----  

-- NV - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros  

* @{
* /  


```

```

/* NV - Register instance definitions */

/* FTFA_FlashConfig */

#define NV_BACKKEY3           NV_BACKKEY3_REG(FTFA_FlashConfig)  

#define NV_BACKKEY2           NV_BACKKEY2_REG(FTFA_FlashConfig)  

#define NV_BACKKEY1           NV_BACKKEY1_REG(FTFA_FlashConfig)  

#define NV_BACKKEY0           NV_BACKKEY0_REG(FTFA_FlashConfig)  

#define NV_BACKKEY7           NV_BACKKEY7_REG(FTFA_FlashConfig)  

#define NV_BACKKEY6           NV_BACKKEY6_REG(FTFA_FlashConfig)

```

```
#define NV_BACKKEY5           NV_BACKKEY5_REG(FTFA_FlashConfig)
#define NV_BACKKEY4           NV_BACKKEY4_REG(FTFA_FlashConfig)
#define NV_FPROT3             NV_FPROT3_REG(FTFA_FlashConfig)
#define NV_FPROT2             NV_FPROT2_REG(FTFA_FlashConfig)
#define NV_FPROT1             NV_FPROT1_REG(FTFA_FlashConfig)
#define NV_FPROT0             NV_FPROT0_REG(FTFA_FlashConfig)
#define NV_FSEC               NV_FSEC_REG(FTFA_FlashConfig)
#define NV_FOPT               NV_FOPT_REG(FTFA_FlashConfig)
```

```
/*!
 * @}
 */
/* /* end of group NV_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* /* end of group NV_Peripheral_Access_Layer */
```

```
/*
-----  
-- OSC Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup OSC_Peripheral_Access_Layer OSC Peripheral Access Layer  
 * @{  
 */
```

```
/** OSC - Register Layout Typedef */
```

```
typedef struct {  
    __IO uint8_t CR;           /*< OSC Control Register, offset: 0x0 */  
} OSC_Type, *OSC_MemMapPtr;
```

```
/* -----  
-- OSC - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros  
* @{  
*/
```

```
/* OSC - Register accessors */  
#define OSC_CR_REG(base) ((base)->CR)
```

```
/*!  
* @}  
* ///* end of group OSC_Register_Accessor_Macros */
```

```
/* -----  
-- OSC Register Masks  
----- */
```

```
/*!  
* @addtogroup OSC_Register_Masks OSC Register Masks  
* @{
```

*/

```
/* CR Bit Fields */

#define OSC_CR_SC16P_MASK          0x1u
#define OSC_CR_SC16P_SHIFT         0
#define OSC_CR_SC16P_WIDTH          1
#define OSC_CR_SC16P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC16P_SHIFT))&OSC_CR_SC16P_MASK)

#define OSC_CR_SC8P_MASK           0x2u
#define OSC_CR_SC8P_SHIFT          1
#define OSC_CR_SC8P_WIDTH          1
#define OSC_CR_SC8P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC8P_SHIFT))&OSC_CR_SC8P_MASK)

#define OSC_CR_SC4P_MASK           0x4u
#define OSC_CR_SC4P_SHIFT          2
#define OSC_CR_SC4P_WIDTH          1
#define OSC_CR_SC4P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC4P_SHIFT))&OSC_CR_SC4P_MASK)

#define OSC_CR_SC2P_MASK           0x8u
#define OSC_CR_SC2P_SHIFT          3
#define OSC_CR_SC2P_WIDTH          1
#define OSC_CR_SC2P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC2P_SHIFT))&OSC_CR_SC2P_MASK)

#define OSC_CR_EREFSTEN_MASK        0x20u
#define OSC_CR_EREFSTEN_SHIFT       5
#define OSC_CR_EREFSTEN_WIDTH       1
#define OSC_CR_EREFSTEN(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_EREFSTEN_SHIFT))&OSC_CR_EREFSTEN_MASK)

#define OSC_CR_ERCLKEN_MASK         0x80u
#define OSC_CR_ERCLKEN_SHIFT        7
```

```
#define OSC_CR_ERCLKEN_WIDTH          1
#define OSC_CR_ERCLKEN(x)
(((uint8_t)((uint8_t)(x)<<OSC_CR_ERCLKEN_SHIFT))&OSC_CR_ERCLKEN_MASK)
```

```
/*!
 * @}
 */
/* /* end of group OSC_Register_Masks */
```

```
/* OSC - Peripheral instance base addresses */
/** Peripheral OSC0 base address */
#define OSC0_BASE          (0x40065000u)
/** Peripheral OSC0 base pointer */
#define OSC0           ((OSC_Type *)OSCO_BASE)
#define OSC0_BASE_PTR      (OSCO)
/** Array initializer of OSC peripheral base addresses */
#define OSC_BASE_ADDRS     { OSC0_BASE }
/** Array initializer of OSC peripheral base pointers */
#define OSC_BASE_PTRS       { OSC0 }
```

```
/*
-----*
-- OSC - Register accessor macros
----- */
```

```
/*
* @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros
* @{
*/

```

```

/* OSC - Register instance definitions */

/* OSC0 */

#define OSC0_CR           OSC_CR_REG(OSC0)

/*
 * @}
 */

/* /* end of group OSC_Register_Accessor_Macros */

/*
 * @}
 */

/* /* end of group OSC_Peripheral_Access_Layer */

/*
-----  

-- PIT Peripheral Access Layer  

----- */

```

```

/*
 * @addtogroup PIT_Peripheral_Access_Layer PIT Peripheral Access Layer
 * @{
 */

/** PIT - Register Layout Typedef */

typedef struct {

    __IO uint32_t MCR;          /**< PIT Module Control Register, offset: 0x0 */

    uint8_t RESERVED_0[220];

    __I uint32_t LTMR64H;       /**< PIT Upper Lifetime Timer Register, offset: 0xE0 */
}

```

```

__I uint32_t LTMR64L;           /**< PIT Lower Lifetime Timer Register, offset: 0xE4 */

uint8_t RESERVED_1[24];

struct {                         /* offset: 0x100, array step: 0x10 */

    __IO uint32_t LDVAL;          /**< Timer Load Value Register, array offset: 0x100, array step:
0x10 */

    __I uint32_t CVAL;           /**< Current Timer Value Register, array offset: 0x104, array step:
0x10 */

    __IO uint32_t TCTRL;          /**< Timer Control Register, array offset: 0x108, array step: 0x10

    __IO uint32_t TFLG;           /**< Timer Flag Register, array offset: 0x10C, array step: 0x10

} CHANNEL[2];

} PIT_Type, *PIT_MemMapPtr;

```

```

/*
-----  

-- PIT - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros  

* @{
* /  


```

```

/* PIT - Register accessors */  

#define PIT_MCR_REG(base)          ((base)->MCR)  

#define PIT_LTMR64H_REG(base)       ((base)->LTMR64H)  

#define PIT_LTMR64L_REG(base)       ((base)->LTMR64L)  

#define PIT_LDVAL_REG(base,index)   ((base)->CHANNEL[index].LDVAL)  

#define PIT_LDVAL_COUNT            2  

#define PIT_CVAL_REG(base,index)    ((base)->CHANNEL[index].CVAL)

```

```
#define PIT_CVAL_COUNT          2
#define PIT_TCTRL_REG(base,index) ((base)->CHANNEL[index].TCTRL)
#define PIT_TCTRL_COUNT          2
#define PIT_TFLG_REG(base,index) ((base)->CHANNEL[index].TFLG)
#define PIT_TFLG_COUNT          2

/*!
 * @}
 */
/* /* end of group PIT_Register_Accessor_Macros */
```

```
/*
-----  
-- PIT Register Masks  
----- */
```

```
/*!  
 * @addtogroup PIT_Register_Masks PIT Register Masks  
 * @{  
 */
```

```
/* MCR Bit Fields */

#define PIT_MCR_FRZ_MASK          0x1u
#define PIT_MCR_FRZ_SHIFT         0
#define PIT_MCR_FRZ_WIDTH         1
#define PIT_MCR_FRZ(x) (((uint32_t)((uint32_t)(x))<<PIT_MCR_FRZ_SHIFT))&PIT_MCR_FRZ_MASK)

#define PIT_MCR_MDIS_MASK          0x2u
#define PIT_MCR_MDIS_SHIFT         1
#define PIT_MCR_MDIS_WIDTH         1
```

```

#define PIT_MCR_MDIS(x)
(((uint32_t) (((uint32_t)(x))<<PIT_MCR_MDIS_SHIFT))&PIT_MCR_MDIS_MASK)

/* LTMR64H Bit Fields */

#define PIT_LTMR64H_LTH_MASK          0xFFFFFFFFu
#define PIT_LTMR64H_LTH_SHIFT         0
#define PIT_LTMR64H_LTH_WIDTH         32
#define PIT_LTMR64H_LTH(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LTMR64H_LTH_SHIFT))&PIT_LTMR64H_LTH_MASK)

/* LTMR64L Bit Fields */

#define PIT_LTMR64L_LTL_MASK          0xFFFFFFFFu
#define PIT_LTMR64L_LTL_SHIFT         0
#define PIT_LTMR64L_LTL_WIDTH         32
#define PIT_LTMR64L_LTL(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LTMR64L_LTL_SHIFT))&PIT_LTMR64L_LTL_MASK)

/* LDVAL Bit Fields */

#define PIT_LDVAL_TSV_MASK           0xFFFFFFFFu
#define PIT_LDVAL_TSV_SHIFT          0
#define PIT_LDVAL_TSV_WIDTH          32
#define PIT_LDVAL_TSV(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LDVAL_TSV_SHIFT))&PIT_LDVAL_TSV_MASK)

/* CVAL Bit Fields */

#define PIT_CVAL_TVL_MASK            0xFFFFFFFFu
#define PIT_CVAL_TVL_SHIFT           0
#define PIT_CVAL_TVL_WIDTH           32
#define PIT_CVAL_TVL(x)
(((uint32_t) (((uint32_t)(x))<<PIT_CVAL_TVL_SHIFT))&PIT_CVAL_TVL_MASK)

/* TCTRL Bit Fields */

#define PIT_TCTRL_TEN_MASK           0x1u
#define PIT_TCTRL_TEN_SHIFT          0
#define PIT_TCTRL_TEN_WIDTH          1

```

```

#define PIT_TCTRL_TEN(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_TEN_SHIFT))&PIT_TCTRL_TEN_MASK)

#define PIT_TCTRL_TIE_MASK          0x2u
#define PIT_TCTRL_TIE_SHIFT         1
#define PIT_TCTRL_TIE_WIDTH         1

#define PIT_TCTRL_TIE(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_TIE_SHIFT))&PIT_TCTRL_TIE_MASK)

#define PIT_TCTRL_CHN_MASK          0x4u
#define PIT_TCTRL_CHN_SHIFT         2
#define PIT_TCTRL_CHN_WIDTH         1

#define PIT_TCTRL_CHN(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_CHN_SHIFT))&PIT_TCTRL_CHN_MASK)

/* TFLG Bit Fields */

#define PIT_TFLG_TIF_MASK          0x1u
#define PIT_TFLG_TIF_SHIFT          0
#define PIT_TFLG_TIF_WIDTH          1

#define PIT_TFLG_TIF(x)
(((uint32_t)((uint32_t)(x)<<PIT_TFLG_TIF_SHIFT))&PIT_TFLG_TIF_MASK)

/*
 */
*! 
* @}

/* /* end of group PIT_Register_Masks */

```

```

/* PIT - Peripheral instance base addresses */

/** Peripheral PIT base address */

#define PIT_BASE                  (0x40037000u)

/** Peripheral PIT base pointer */

#define PIT                      ((PIT_Type *)PIT_BASE)

#define PIT_BASE_PTR             (PIT)

```

```

/** Array initializer of PIT peripheral base addresses */
#define PIT_BASE_ADDRS          { PIT_BASE }

/** Array initializer of PIT peripheral base pointers */
#define PIT_BASE_PTRS           { PIT }

/*
-----  

-- PIT - Register accessor macros  

----- */
/*!  

* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros  

* @{
* /  

  

/* PIT - Register instance definitions */  

/* PIT */  

#define PIT_MCR                  PIT_MCR_REG(PIT)  

#define PIT_LTMR64H               PIT_LTMR64H_REG(PIT)  

#define PIT_LTMR64L               PIT_LTMR64L_REG(PIT)  

#define PIT_LDVAL0                PIT_LDVAL_REG(PIT,0)  

#define PIT_CVAL0                 PIT_CVAL_REG(PIT,0)  

#define PIT_TCTRL0                PIT_TCTRL_REG(PIT,0)  

#define PIT_TFLG0                 PIT_TFLG_REG(PIT,0)  

#define PIT_LDVAL1                PIT_LDVAL_REG(PIT,1)  

#define PIT_CVAL1                 PIT_CVAL_REG(PIT,1)  

#define PIT_TCTRL1                PIT_TCTRL_REG(PIT,1)  

#define PIT_TFLG1                 PIT_TFLG_REG(PIT,1)

```

```
/* PIT - Register array accessors */

#define PIT_LDVAL(index)          PIT_LDVAL_REG(PIT,index)
#define PIT_CVAL(index)           PIT_CVAL_REG(PIT,index)
#define PIT_TCTRL(index)          PIT_TCTRL_REG(PIT,index)
#define PIT_TFLG(index)           PIT_TFLG_REG(PIT,index)
```

```
/*
 * @}
 */
/* end of group PIT_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group PIT_Peripheral_Access_Layer */
```

```
/*
-----  
-- PMC Peripheral Access Layer  
----- */
```

```
/*
 * @addtogroup PMC_Peripheral_Access_Layer PMC Peripheral Access Layer
 * @{
 */

```

```
/** PMC - Register Layout Typedef */

typedef struct {
    _IO uint8_t LVDSC1;           /**< Low Voltage Detect Status And Control 1 register, offset: 0x0
*/
```

```
    __IO uint8_t LVDSC2;           /**< Low Voltage Detect Status And Control 2 register, offset: 0x1
*/
    __IO uint8_t REGSC;           /**< Regulator Status And Control register, offset: 0x2 */
} PMC_Type, *PMC_MemMapPtr;
```

```
/*
-----*
-- PMC - Register accessor macros
----- */
```

```
/*!
* @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros
* @{
*/

```

```
/* PMC - Register accessors */
#define PMC_LVDSC1_REG(base)      ((base)->LVDSC1)
#define PMC_LVDSC2_REG(base)      ((base)->LVDSC2)
#define PMC_REGSC_REG(base)       ((base)->REGSC)
```

```
/*!
* @}
*/
/* /* end of group PMC_Register_Accessor_Macros */
```

```
/*
-----*
-- PMC Register Masks
----- */
```

```

/*!

* @addtogroup PMC_Register_Masks PMC Register Masks

* @{

*/



/* LVDSC1 Bit Fields */

#define PMC_LVDSC1_LVDV_MASK          0x3u
#define PMC_LVDSC1_LVDV_SHIFT         0
#define PMC_LVDSC1_LVDV_WIDTH         2
#define PMC_LVDSC1_LVDV(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDV_SHIFT))&PMC_LVDSC1_LVDV_MASK)
#define PMC_LVDSC1_LVDRE_MASK         0x10u
#define PMC_LVDSC1_LVDRE_SHIFT        4
#define PMC_LVDSC1_LVDRE_WIDTH        1
#define PMC_LVDSC1_LVDRE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDRE_SHIFT))&PMC_LVDSC1_LVDRE_MASK)
#define PMC_LVDSC1_LVDIE_MASK         0x20u
#define PMC_LVDSC1_LVDIE_SHIFT        5
#define PMC_LVDSC1_LVDIE_WIDTH        1
#define PMC_LVDSC1_LVDIE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDIE_SHIFT))&PMC_LVDSC1_LVDIE_MASK)
#define PMC_LVDSC1_LVDACK_MASK        0x40u
#define PMC_LVDSC1_LVDACK_SHIFT       6
#define PMC_LVDSC1_LVDACK_WIDTH       1
#define PMC_LVDSC1_LVDACK(x)          (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDACK_SHIFT))&PMC_LVDSC1_LVDACK_MASK)
#define PMC_LVDSC1_LVDF_MASK          0x80u
#define PMC_LVDSC1_LVDF_SHIFT         7
#define PMC_LVDSC1_LVDF_WIDTH         1

```

```

#define PMC_LVDSC1_LVDF(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC1_LVDF_SHIFT))&PMC_LVDSC1_LVDF_MASK)

/* LVDSC2 Bit Fields */

#define PMC_LVDSC2_LVWV_MASK          0x3u
#define PMC_LVDSC2_LVWV_SHIFT         0
#define PMC_LVDSC2_LVWV_WIDTH         2
#define PMC_LVDSC2_LVWV(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWV_SHIFT))&PMC_LVDSC2_LVWV_MASK)

#define PMC_LVDSC2_LVWIE_MASK         0x20u
#define PMC_LVDSC2_LVWIE_SHIFT        5
#define PMC_LVDSC2_LVWIE_WIDTH        1
#define PMC_LVDSC2_LVWIE(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWIE_SHIFT))&PMC_LVDSC2_LVWIE_MASK)

#define PMC_LVDSC2_LVWACK_MASK        0x40u
#define PMC_LVDSC2_LVWACK_SHIFT       6
#define PMC_LVDSC2_LVWACK_WIDTH       1
#define PMC_LVDSC2_LVWACK(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWACK_SHIFT))&PMC_LVDSC2_LVWACK_MASK)

#define PMC_LVDSC2_LVWF_MASK          0x80u
#define PMC_LVDSC2_LVWF_SHIFT         7
#define PMC_LVDSC2_LVWF_WIDTH         1
#define PMC_LVDSC2_LVWF(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWF_SHIFT))&PMC_LVDSC2_LVWF_MASK)

/* REGSC Bit Fields */

#define PMC_REGSC_BGBE_MASK           0x1u
#define PMC_REGSC_BGBE_SHIFT          0
#define PMC_REGSC_BGBE_WIDTH          1
#define PMC_REGSC_BGBE(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGBE_SHIFT))&PMC_REGSC_BGBE_MASK)

#define PMC_REGSC_REGONS_MASK         0x4u
#define PMC_REGSC_REGONS_SHIFT        2

```

```

#define PMC_REGSC_REGONS_WIDTH          1
#define PMC_REGSC_REGONS(x)             (((uint8_t)((uint8_t)(x)<<PMC_REGSC_REGONS_SHIFT))&PMC_REGSC_REGONS_MASK)
#define PMC_REGSC_ACKISO_MASK           0x8u
#define PMC_REGSC_ACKISO_SHIFT          3
#define PMC_REGSC_ACKISO_WIDTH          1
#define PMC_REGSC_ACKISO(x)             (((uint8_t)((uint8_t)(x)<<PMC_REGSC_ACKISO_SHIFT))&PMC_REGSC_ACKISO_MASK)
#define PMC_REGSC_BGEN_MASK             0x10u
#define PMC_REGSC_BGEN_SHIFT            4
#define PMC_REGSC_BGEN_WIDTH            1
#define PMC_REGSC_BGEN(x)               (((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGEN_SHIFT))&PMC_REGSC_BGEN_MASK)

```

```

/*
 * @}
 */
/* end of group PMC_Register_Masks */

```

```

/* PMC - Peripheral instance base addresses */
/** Peripheral PMC base address */
#define PMC_BASE                      (0x4007D000u)
/** Peripheral PMC base pointer */
#define PMC                         ((PMC_Type *)PMC_BASE)
#define PMC_BASE_PTR                  (PMC)
/** Array initializer of PMC peripheral base addresses */
#define PMC_BASE_ADDRS                { PMC_BASE }
/** Array initializer of PMC peripheral base pointers */
#define PMC_BASE_PTRS                 { PMC }

```

```
/*
-- PMC - Register accessor macros
----- */
/*!
* @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros
* @{
*/
/* PMC - Register instance definitions */
/* PMC */
#define PMC_LVDSC1          PMC_LVDSC1_REG(PMC)
#define PMC_LVDSC2          PMC_LVDSC2_REG(PMC)
#define PMC_REGSC           PMC_REGSC_REG(PMC)

/*!
* @}
*/
/* end of group PMC_Register_Accessor_Macros */

/*!
* @}
*/
/* end of group PMC_Peripheral_Access_Layer */

/*
-- PORT Peripheral Access Layer
----- */

```

```

/*!

* @addtogroup PORT_Peripheral_Access_Layer PORT Peripheral Access Layer

* @{

*/



/** PORT - Register Layout Typedef */

typedef struct {

    __IO uint32_t PCR[32];           /**< Pin Control Register n, array offset: 0x0, array step: 0x4 */

    __O uint32_t GPCLR;             /**< Global Pin Control Low Register, offset: 0x80 */

    __O uint32_t GPCHR;             /**< Global Pin Control High Register, offset: 0x84 */

    uint8_t RESERVED_0[24];

    __IO uint32_t ISFR;             /**< Interrupt Status Flag Register, offset: 0xA0 */

} PORT_Type, *PORT_MemMapPtr;

```

```

/*
-----  

-- PORT - Register accessor macros  

----- */

```

```

/*!

* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros

* @{

*/




```

```

/* PORT - Register accessors */

#define PORT_PCR_REG(base,index)      ((base)->PCR[index])

#define PORT_PCR_COUNT                32

#define PORT_GPCLR_REG(base)          ((base)->GPCLR)

```

```
#define PORT_GPCHR_REG(base)          ((base)->GPCHR)
#define PORT_ISFR_REG(base)           ((base)->ISFR)

/*!
 * @}
 */
/* /* end of group PORT_Register_Accessor_Macros */
```

```
/*
-----  
-- PORT Register Masks  
----- */
```

```
/*!  
 * @addtogroup PORT_Register_Masks PORT Register Masks  
 * @{  
 */
```

```
/* PCR Bit Fields */

#define PORT_PCR_PS_MASK          0x1u
#define PORT_PCR_PS_SHIFT         0
#define PORT_PCR_PS_WIDTH         1
#define PORT_PCR_PS(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_PS_SHIFT))&PORT_PCR_PS_MASK

#define PORT_PCR_PE_MASK          0x2u
#define PORT_PCR_PE_SHIFT         1
#define PORT_PCR_PE_WIDTH         1
#define PORT_PCR_PE(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_PE_SHIFT))&PORT_PCR_PE_MASK

#define PORT_PCR_SRE_MASK          0x4u
```

```

#define PORT_PCR_SRE_SHIFT          2
#define PORT_PCR_SRE_WIDTH          1
#define PORT_PCR_SRE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_SRE_SHIFT))&PORT_PCR_SRE_MASK)
#define PORT_PCR_PFE_MASK           0x10u
#define PORT_PCR_PFE_SHIFT          4
#define PORT_PCR_PFE_WIDTH          1
#define PORT_PCR_PFE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_PFE_SHIFT))&PORT_PCR_PFE_MASK)
#define PORT_PCR_DSE_MASK           0x40u
#define PORT_PCR_DSE_SHIFT          6
#define PORT_PCR_DSE_WIDTH          1
#define PORT_PCR_DSE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_DSE_SHIFT))&PORT_PCR_DSE_MASK)
#define PORT_PCR_MUX_MASK           0x700u
#define PORT_PCR_MUX_SHIFT          8
#define PORT_PCR_MUX_WIDTH          3
#define PORT_PCR_MUX(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_MUX_SHIFT))&PORT_PCR_MUX_MASK)
#define PORT_PCR_IRQC_MASK           0xF0000u
#define PORT_PCR_IRQC_SHIFT          16
#define PORT_PCR_IRQC_WIDTH          4
#define PORT_PCR_IRQC(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_IRQC_SHIFT))&PORT_PCR_IRQC_MASK)
#define PORT_PCR_ISF_MASK            0x1000000u
#define PORT_PCR_ISF_SHIFT           24
#define PORT_PCR_ISF_WIDTH           1
#define PORT_PCR_ISF(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_ISF_SHIFT))&PORT_PCR_ISF_MASK)
/* GPCLR Bit Fields */
#define PORT_GPCLR_GPWD_MASK         0xFFFFu

```

```

#define PORT_GPCLR_GPWD_SHIFT          0
#define PORT_GPCLR_GPWD_WIDTH          16
#define PORT_GPCLR_GPWD(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWD_SHIFT))&PORT_GPCLR_GPWD_MASK)
#define PORT_GPCLR_GPWE_MASK           0xFFFF0000u
#define PORT_GPCLR_GPWE_SHIFT          16
#define PORT_GPCLR_GPWE_WIDTH          16
#define PORT_GPCLR_GPWE(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWE_SHIFT))&PORT_GPCLR_GPWE_MASK)

/* GPCHR Bit Fields */

#define PORT_GPCHR_GPWD_MASK           0xFFFFu
#define PORT_GPCHR_GPWD_SHIFT          0
#define PORT_GPCHR_GPWD_WIDTH          16
#define PORT_GPCHR_GPWD(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWD_SHIFT))&PORT_GPCHR_GPWD_MASK)
#define PORT_GPCHR_GPWE_MASK           0xFFFF0000u
#define PORT_GPCHR_GPWE_SHIFT          16
#define PORT_GPCHR_GPWE_WIDTH          16
#define PORT_GPCHR_GPWE(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWE_SHIFT))&PORT_GPCHR_GPWE_MASK)

/* ISFR Bit Fields */

#define PORT_ISFR_ISF_MASK             0xFFFFFFFFu
#define PORT_ISFR_ISF_SHIFT            0
#define PORT_ISFR_ISF_WIDTH            32
#define PORT_ISFR_ISF(x)               (((uint32_t)((uint32_t)(x)<<PORT_ISFR_ISF_SHIFT))&PORT_ISFR_ISF_MASK)

/*
 * @}
 */

/* /* end of group PORT_Register_Masks */

```

```

/* PORT - Peripheral instance base addresses */

/** Peripheral PORTA base address */

#define PORTA_BASE          (0x40049000u)

/** Peripheral PORTA base pointer */

#define PORTA          ((PORT_Type *)PORTA_BASE)

#define PORTA_BASE_PTR      (PORTA)

/** Peripheral PORTB base address */

#define PORTB_BASE          (0x4004A000u)

/** Peripheral PORTB base pointer */

#define PORTB          ((PORT_Type *)PORTB_BASE)

#define PORTB_BASE_PTR      (PORTB)

/** Peripheral PORTC base address */

#define PORTC_BASE          (0x4004B000u)

/** Peripheral PORTC base pointer */

#define PORTC          ((PORT_Type *)PORTC_BASE)

#define PORTC_BASE_PTR      (PORTC)

/** Peripheral PORTD base address */

#define PORTD_BASE          (0x4004C000u)

/** Peripheral PORTD base pointer */

#define PORTD          ((PORT_Type *)PORTD_BASE)

#define PORTD_BASE_PTR      (PORTD)

/** Peripheral PORTE base address */

#define PORTE_BASE          (0x4004D000u)

/** Peripheral PORTE base pointer */

#define PORTE          ((PORT_Type *)PORTE_BASE)

#define PORTE_BASE_PTR      (PORTE)

/** Array initializer of PORT peripheral base addresses */

```

```

#define PORT_BASE_ADDRS          { PORTA_BASE, PORTB_BASE, PORTC_BASE, PORTD_BASE,
PORTE_BASE }

/** Array initializer of PORT peripheral base pointers */

#define PORT_BASE_PTRS           { PORTA, PORTB, PORTC, PORTD, PORTE }

/*
-----*
-- PORT - Register accessor macros
----- */
/*!
* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros
* @{
*/
/* PORT - Register instance definitions */
/* PORTA */

#define PORTA_PCR0                PORT_PCR_REG(PORTA,0)
#define PORTA_PCR1                PORT_PCR_REG(PORTA,1)
#define PORTA_PCR2                PORT_PCR_REG(PORTA,2)
#define PORTA_PCR3                PORT_PCR_REG(PORTA,3)
#define PORTA_PCR4                PORT_PCR_REG(PORTA,4)
#define PORTA_PCR5                PORT_PCR_REG(PORTA,5)
#define PORTA_PCR6                PORT_PCR_REG(PORTA,6)
#define PORTA_PCR7                PORT_PCR_REG(PORTA,7)
#define PORTA_PCR8                PORT_PCR_REG(PORTA,8)
#define PORTA_PCR9                PORT_PCR_REG(PORTA,9)
#define PORTA_PCR10               PORT_PCR_REG(PORTA,10)
#define PORTA_PCR11               PORT_PCR_REG(PORTA,11)

```

```
#define PORTA_PCR12          PORT_PCR_REG(PORTA,12)
#define PORTA_PCR13          PORT_PCR_REG(PORTA,13)
#define PORTA_PCR14          PORT_PCR_REG(PORTA,14)
#define PORTA_PCR15          PORT_PCR_REG(PORTA,15)
#define PORTA_PCR16          PORT_PCR_REG(PORTA,16)
#define PORTA_PCR17          PORT_PCR_REG(PORTA,17)
#define PORTA_PCR18          PORT_PCR_REG(PORTA,18)
#define PORTA_PCR19          PORT_PCR_REG(PORTA,19)
#define PORTA_PCR20          PORT_PCR_REG(PORTA,20)
#define PORTA_PCR21          PORT_PCR_REG(PORTA,21)
#define PORTA_PCR22          PORT_PCR_REG(PORTA,22)
#define PORTA_PCR23          PORT_PCR_REG(PORTA,23)
#define PORTA_PCR24          PORT_PCR_REG(PORTA,24)
#define PORTA_PCR25          PORT_PCR_REG(PORTA,25)
#define PORTA_PCR26          PORT_PCR_REG(PORTA,26)
#define PORTA_PCR27          PORT_PCR_REG(PORTA,27)
#define PORTA_PCR28          PORT_PCR_REG(PORTA,28)
#define PORTA_PCR29          PORT_PCR_REG(PORTA,29)
#define PORTA_PCR30          PORT_PCR_REG(PORTA,30)
#define PORTA_PCR31          PORT_PCR_REG(PORTA,31)
#define PORTA_GPCLR          PORT_GPCLR_REG(PORTA)
#define PORTA_GPCHR          PORT_GPCHR_REG(PORTA)
#define PORTA_ISFR           PORT_ISFR_REG(PORTA)

/* PORTB */

#define PORTB_PCR0           PORT_PCR_REG(PORTB,0)
#define PORTB_PCR1           PORT_PCR_REG(PORTB,1)
#define PORTB_PCR2           PORT_PCR_REG(PORTB,2)
#define PORTB_PCR3           PORT_PCR_REG(PORTB,3)
#define PORTB_PCR4           PORT_PCR_REG(PORTB,4)
```

#define PORTB_PCR5	PORT_PCR_REG(PORTB,5)
#define PORTB_PCR6	PORT_PCR_REG(PORTB,6)
#define PORTB_PCR7	PORT_PCR_REG(PORTB,7)
#define PORTB_PCR8	PORT_PCR_REG(PORTB,8)
#define PORTB_PCR9	PORT_PCR_REG(PORTB,9)
#define PORTB_PCR10	PORT_PCR_REG(PORTB,10)
#define PORTB_PCR11	PORT_PCR_REG(PORTB,11)
#define PORTB_PCR12	PORT_PCR_REG(PORTB,12)
#define PORTB_PCR13	PORT_PCR_REG(PORTB,13)
#define PORTB_PCR14	PORT_PCR_REG(PORTB,14)
#define PORTB_PCR15	PORT_PCR_REG(PORTB,15)
#define PORTB_PCR16	PORT_PCR_REG(PORTB,16)
#define PORTB_PCR17	PORT_PCR_REG(PORTB,17)
#define PORTB_PCR18	PORT_PCR_REG(PORTB,18)
#define PORTB_PCR19	PORT_PCR_REG(PORTB,19)
#define PORTB_PCR20	PORT_PCR_REG(PORTB,20)
#define PORTB_PCR21	PORT_PCR_REG(PORTB,21)
#define PORTB_PCR22	PORT_PCR_REG(PORTB,22)
#define PORTB_PCR23	PORT_PCR_REG(PORTB,23)
#define PORTB_PCR24	PORT_PCR_REG(PORTB,24)
#define PORTB_PCR25	PORT_PCR_REG(PORTB,25)
#define PORTB_PCR26	PORT_PCR_REG(PORTB,26)
#define PORTB_PCR27	PORT_PCR_REG(PORTB,27)
#define PORTB_PCR28	PORT_PCR_REG(PORTB,28)
#define PORTB_PCR29	PORT_PCR_REG(PORTB,29)
#define PORTB_PCR30	PORT_PCR_REG(PORTB,30)
#define PORTB_PCR31	PORT_PCR_REG(PORTB,31)
#define PORTB_GPCLR	PORT_GPCLR_REG(PORTB)
#define PORTB_GPCHR	PORT_GPCHR_REG(PORTB)

```
#define PORTB_ISFR          PORT_ISFR_REG(PORTB)
/* PORTC */
#define PORTC_PCR0            PORT_PCR_REG(PORTC,0)
#define PORTC_PCR1            PORT_PCR_REG(PORTC,1)
#define PORTC_PCR2            PORT_PCR_REG(PORTC,2)
#define PORTC_PCR3            PORT_PCR_REG(PORTC,3)
#define PORTC_PCR4            PORT_PCR_REG(PORTC,4)
#define PORTC_PCR5            PORT_PCR_REG(PORTC,5)
#define PORTC_PCR6            PORT_PCR_REG(PORTC,6)
#define PORTC_PCR7            PORT_PCR_REG(PORTC,7)
#define PORTC_PCR8            PORT_PCR_REG(PORTC,8)
#define PORTC_PCR9            PORT_PCR_REG(PORTC,9)
#define PORTC_PCR10           PORT_PCR_REG(PORTC,10)
#define PORTC_PCR11           PORT_PCR_REG(PORTC,11)
#define PORTC_PCR12           PORT_PCR_REG(PORTC,12)
#define PORTC_PCR13           PORT_PCR_REG(PORTC,13)
#define PORTC_PCR14           PORT_PCR_REG(PORTC,14)
#define PORTC_PCR15           PORT_PCR_REG(PORTC,15)
#define PORTC_PCR16           PORT_PCR_REG(PORTC,16)
#define PORTC_PCR17           PORT_PCR_REG(PORTC,17)
#define PORTC_PCR18           PORT_PCR_REG(PORTC,18)
#define PORTC_PCR19           PORT_PCR_REG(PORTC,19)
#define PORTC_PCR20           PORT_PCR_REG(PORTC,20)
#define PORTC_PCR21           PORT_PCR_REG(PORTC,21)
#define PORTC_PCR22           PORT_PCR_REG(PORTC,22)
#define PORTC_PCR23           PORT_PCR_REG(PORTC,23)
#define PORTC_PCR24           PORT_PCR_REG(PORTC,24)
#define PORTC_PCR25           PORT_PCR_REG(PORTC,25)
#define PORTC_PCR26           PORT_PCR_REG(PORTC,26)
```

```
#define PORTC_PCR27          PORT_PCR_REG(PORTC,27)
#define PORTC_PCR28          PORT_PCR_REG(PORTC,28)
#define PORTC_PCR29          PORT_PCR_REG(PORTC,29)
#define PORTC_PCR30          PORT_PCR_REG(PORTC,30)
#define PORTC_PCR31          PORT_PCR_REG(PORTC,31)
#define PORTC_GPCLR          PORT_GPCLR_REG(PORTC)
#define PORTC_GPCHR          PORT_GPCHR_REG(PORTC)
#define PORTC_ISFR           PORT_ISFR_REG(PORTC)

/* PORTD */

#define PORTD_PCR0           PORT_PCR_REG(PORTD,0)
#define PORTD_PCR1           PORT_PCR_REG(PORTD,1)
#define PORTD_PCR2           PORT_PCR_REG(PORTD,2)
#define PORTD_PCR3           PORT_PCR_REG(PORTD,3)
#define PORTD_PCR4           PORT_PCR_REG(PORTD,4)
#define PORTD_PCR5           PORT_PCR_REG(PORTD,5)
#define PORTD_PCR6           PORT_PCR_REG(PORTD,6)
#define PORTD_PCR7           PORT_PCR_REG(PORTD,7)
#define PORTD_PCR8           PORT_PCR_REG(PORTD,8)
#define PORTD_PCR9           PORT_PCR_REG(PORTD,9)
#define PORTD_PCR10          PORT_PCR_REG(PORTD,10)
#define PORTD_PCR11          PORT_PCR_REG(PORTD,11)
#define PORTD_PCR12          PORT_PCR_REG(PORTD,12)
#define PORTD_PCR13          PORT_PCR_REG(PORTD,13)
#define PORTD_PCR14          PORT_PCR_REG(PORTD,14)
#define PORTD_PCR15          PORT_PCR_REG(PORTD,15)
#define PORTD_PCR16          PORT_PCR_REG(PORTD,16)
#define PORTD_PCR17          PORT_PCR_REG(PORTD,17)
#define PORTD_PCR18          PORT_PCR_REG(PORTD,18)
#define PORTD_PCR19          PORT_PCR_REG(PORTD,19)
```

```

#define PORTD_PCR20      PORT_PCR_REG(PORTD,20)
#define PORTD_PCR21      PORT_PCR_REG(PORTD,21)
#define PORTD_PCR22      PORT_PCR_REG(PORTD,22)
#define PORTD_PCR23      PORT_PCR_REG(PORTD,23)
#define PORTD_PCR24      PORT_PCR_REG(PORTD,24)
#define PORTD_PCR25      PORT_PCR_REG(PORTD,25)
#define PORTD_PCR26      PORT_PCR_REG(PORTD,26)
#define PORTD_PCR27      PORT_PCR_REG(PORTD,27)
#define PORTD_PCR28      PORT_PCR_REG(PORTD,28)
#define PORTD_PCR29      PORT_PCR_REG(PORTD,29)
#define PORTD_PCR30      PORT_PCR_REG(PORTD,30)
#define PORTD_PCR31      PORT_PCR_REG(PORTD,31)
#define PORTD_GPCLR      PORT_GPCLR_REG(PORTD)
#define PORTD_GPCHR      PORT_GPCHR_REG(PORTD)
#define PORTD_ISFR       PORT_ISFR_REG(PORTD)

/* PORTE */

#define PORTE_PCR0       PORT_PCR_REG(PORTE,0)
#define PORTE_PCR1       PORT_PCR_REG(PORTE,1)
#define PORTE_PCR2       PORT_PCR_REG(PORTE,2)
#define PORTE_PCR3       PORT_PCR_REG(PORTE,3)
#define PORTE_PCR4       PORT_PCR_REG(PORTE,4)
#define PORTE_PCR5       PORT_PCR_REG(PORTE,5)
#define PORTE_PCR6       PORT_PCR_REG(PORTE,6)
#define PORTE_PCR7       PORT_PCR_REG(PORTE,7)
#define PORTE_PCR8       PORT_PCR_REG(PORTE,8)
#define PORTE_PCR9       PORT_PCR_REG(PORTE,9)
#define PORTE_PCR10      PORT_PCR_REG(PORTE,10)
#define PORTE_PCR11      PORT_PCR_REG(PORTE,11)
#define PORTE_PCR12      PORT_PCR_REG(PORTE,12)

```

#define PORTE_PCR13	PORT_PCR_REG(PORTE,13)
#define PORTE_PCR14	PORT_PCR_REG(PORTE,14)
#define PORTE_PCR15	PORT_PCR_REG(PORTE,15)
#define PORTE_PCR16	PORT_PCR_REG(PORTE,16)
#define PORTE_PCR17	PORT_PCR_REG(PORTE,17)
#define PORTE_PCR18	PORT_PCR_REG(PORTE,18)
#define PORTE_PCR19	PORT_PCR_REG(PORTE,19)
#define PORTE_PCR20	PORT_PCR_REG(PORTE,20)
#define PORTE_PCR21	PORT_PCR_REG(PORTE,21)
#define PORTE_PCR22	PORT_PCR_REG(PORTE,22)
#define PORTE_PCR23	PORT_PCR_REG(PORTE,23)
#define PORTE_PCR24	PORT_PCR_REG(PORTE,24)
#define PORTE_PCR25	PORT_PCR_REG(PORTE,25)
#define PORTE_PCR26	PORT_PCR_REG(PORTE,26)
#define PORTE_PCR27	PORT_PCR_REG(PORTE,27)
#define PORTE_PCR28	PORT_PCR_REG(PORTE,28)
#define PORTE_PCR29	PORT_PCR_REG(PORTE,29)
#define PORTE_PCR30	PORT_PCR_REG(PORTE,30)
#define PORTE_PCR31	PORT_PCR_REG(PORTE,31)
#define PORTE_GPCLR	PORT_GPCLR_REG(PORTE)
#define PORTE_GPCHR	PORT_GPCHR_REG(PORTE)
#define PORTE_ISFR	PORT_ISFR_REG(PORTE)

/* PORT - Register array accessors */

#define PORTA_PCR(index)	PORT_PCR_REG(PORTA,index)
#define PORTB_PCR(index)	PORT_PCR_REG(PORTB,index)
#define PORTC_PCR(index)	PORT_PCR_REG(PORTC,index)
#define PORTD_PCR(index)	PORT_PCR_REG(PORTD,index)
#define PORTE_PCR(index)	PORT_PCR_REG(PORTE,index)

```
/*!  
 * @}  
 */ /* end of group PORT_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group PORT_Peripheral_Access_Layer */
```

```
/* -----  
-- RCM Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup RCM_Peripheral_Access_Layer RCM Peripheral Access Layer  
 * @{  
 */
```

```
/** RCM - Register Layout Typedef */  
  
typedef struct {  
    __I uint8_t SRS0;           /**< System Reset Status Register 0, offset: 0x0 */  
    __I uint8_t SRS1;           /**< System Reset Status Register 1, offset: 0x1 */  
    uint8_t RESERVED_0[2];  
    __IO uint8_t RPFC;          /**< Reset Pin Filter Control register, offset: 0x4 */  
    __IO uint8_t RPFW;          /**< Reset Pin Filter Width register, offset: 0x5 */  
} RCM_Type, *RCM_MemMapPtr;
```

```
/* -----
-- RCM - Register accessor macros
----- */

/*!
* @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros
* @{
*
/* RCM - Register accessors */
#define RCM_SRS0_REG(base) ((base)->SRS0)
#define RCM_SRS1_REG(base) ((base)->SRS1)
#define RCM_RPFC_REG(base) ((base)->RPFC)
#define RCM_RPFW_REG(base) ((base)->RPFW)

/*!
* @}
*/
/* end of group RCM_Register_Accessor_Macros */
```

```
/* -----
-- RCM Register Masks
----- */

/*!
* @addtogroup RCM_Register_Masks RCM Register Masks
* @{
*
*/
```

```

/* SRS0 Bit Fields */

#define RCM_SRS0_WAKEUP_MASK          0x1u
#define RCM_SRS0_WAKEUP_SHIFT         0
#define RCM_SRS0_WAKEUP_WIDTH         1
#define RCM_SRS0_WAKEUP(x)            (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WAKEUP_SHIFT))&RCM_SRS0_WAKEUP_MASK)

#define RCM_SRS0_LVD_MASK            0x2u
#define RCM_SRS0_LVD_SHIFT           1
#define RCM_SRS0_LVD_WIDTH           1
#define RCM_SRS0_LVD(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LVD_SHIFT))&RCM_SRS0_LVD_MASK)

#define RCM_SRS0_LOC_MASK            0x4u
#define RCM_SRS0_LOC_SHIFT           2
#define RCM_SRS0_LOC_WIDTH           1
#define RCM_SRS0_LOC(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOC_SHIFT))&RCM_SRS0_LOC_MASK)

#define RCM_SRS0_LOL_MASK            0x8u
#define RCM_SRS0_LOL_SHIFT           3
#define RCM_SRS0_LOL_WIDTH           1
#define RCM_SRS0_LOL(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOL_SHIFT))&RCM_SRS0_LOL_MASK)

#define RCM_SRS0_WDOG_MASK           0x20u
#define RCM_SRS0_WDOG_SHIFT          5
#define RCM_SRS0_WDOG_WIDTH          1
#define RCM_SRS0_WDOG(x)             (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WDOG_SHIFT))&RCM_SRS0_WDOG_MASK)

#define RCM_SRS0_PIN_MASK            0x40u
#define RCM_SRS0_PIN_SHIFT           6
#define RCM_SRS0_PIN_WIDTH           1

```

```

#define RCM_SRS0_PIN(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS0_PIN_SHIFT))&RCM_SRS0_PIN_MASK)

#define RCM_SRS0_POR_MASK          0x80u
#define RCM_SRS0_POR_SHIFT         7
#define RCM_SRS0_POR_WIDTH         1

#define RCM_SRS0_POR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS0_POR_SHIFT))&RCM_SRS0_POR_MASK)

/* SRS1 Bit Fields */

#define RCM_SRS1_LOCKUP_MASK       0x2u
#define RCM_SRS1_LOCKUP_SHIFT      1
#define RCM_SRS1_LOCKUP_WIDTH      1

#define RCM_SRS1_LOCKUP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_LOCKUP_SHIFT))&RCM_SRS1_LOCKUP_MASK)

#define RCM_SRS1_SW_MASK           0x4u
#define RCM_SRS1_SW_SHIFT          2
#define RCM_SRS1_SW_WIDTH          1

#define RCM_SRS1_SW(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SW_SHIFT))&RCM_SRS1_SW_MASK)

#define RCM_SRS1_MDM_AP_MASK        0x8u
#define RCM_SRS1_MDM_AP_SHIFT       3
#define RCM_SRS1_MDM_AP_WIDTH       1

#define RCM_SRS1_MDM_AP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_MDM_AP_SHIFT))&RCM_SRS1_MDM_AP_MASK)

#define RCM_SRS1_SACKERR_MASK       0x20u
#define RCM_SRS1_SACKERR_SHIFT      5
#define RCM_SRS1_SACKERR_WIDTH      1

#define RCM_SRS1_SACKERR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SACKERR_SHIFT))&RCM_SRS1_SACKERR_MASK)

/* RPFC Bit Fields */

#define RCM_RPFC_RSTFLTSRW_MASK     0x3u
#define RCM_RPFC_RSTFLTSRW_SHIFT     0

```

```

#define RCM_RPFC_RSTFLTSRW_WIDTH          2
#define RCM_RPFC_RSTFLTSRW(x) \
(((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSRW_SHIFT))&RCM_RPFC_RSTFLTSRW_MASK)

#define RCM_RPFC_RSTFLTSS_MASK            0x4u
#define RCM_RPFC_RSTFLTSS_SHIFT          2
#define RCM_RPFC_RSTFLTSS_WIDTH          1
#define RCM_RPFC_RSTFLTSS(x) \
(((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSS_SHIFT))&RCM_RPFC_RSTFLTSS_MASK)

/* RPFW Bit Fields */

#define RCM_RPFW_RSTFLTSEL_MASK          0x1Fu
#define RCM_RPFW_RSTFLTSEL_SHIFT         0
#define RCM_RPFW_RSTFLTSEL_WIDTH         5
#define RCM_RPFW_RSTFLTSEL(x) \
(((uint8_t)((uint8_t)(x)<<RCM_RPFW_RSTFLTSEL_SHIFT))&RCM_RPFW_RSTFLTSEL_MASK)

/*
 */

```

```

/* RCM - Peripheral instance base addresses */

/** Peripheral RCM base address */
#define RCM_BASE                      (0x4007F000u)

/** Peripheral RCM base pointer */
#define RCM                         ((RCM_Type *)RCM_BASE)
#define RCM_BASE_PTR                  (RCM)

/** Array initializer of RCM peripheral base addresses */
#define RCM_BASE_ADDRS                { RCM_BASE }

/** Array initializer of RCM peripheral base pointers */
#define RCM_BASE_PTRS                 { RCM }

```

```
/*
-----  
-- RCM - Register accessor macros  
----- */  
  
/*!  
 * @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros  
 * @{  
 */  
  
/* RCM - Register instance definitions */  
/* RCM */  
  
#define RCM_SRS0          RCM_SRS0_REG(RCM)  
#define RCM_SRS1          RCM_SRS1_REG(RCM)  
#define RCM_RPFC          RCM_RPFC_REG(RCM)  
#define RCM_RPFW          RCM_RPFW_REG(RCM)  
  
/*!  
 * @}  
 */ /* end of group RCM_Register_Accessor_Macros */  
  
/*!  
 * @}  
 */ /* end of group RCM_Peripheral_Access_Layer */  
  
/*
-----
```

```

-- ROM Peripheral Access Layer
----- */
/*!
* @addtogroup ROM_Peripheral_Access_Layer ROM Peripheral Access Layer
* @{
*/
/** ROM - Register Layout Typedef */
typedef struct {
    __I uint32_t ENTRY[3];           /**< Entry, array offset: 0x0, array step: 0x4 */
    __I uint32_t TABLEMARK;          /**< End of Table Marker Register, offset: 0xC */
    uint8_t RESERVED_0[4028];
    __I uint32_t SYSACCESS;          /**< System Access Register, offset: 0xFCC */
    __I uint32_t PERIPHID4;          /**< Peripheral ID Register, offset: 0xFD0 */
    __I uint32_t PERIPHID5;          /**< Peripheral ID Register, offset: 0xFD4 */
    __I uint32_t PERIPHID6;          /**< Peripheral ID Register, offset: 0xFD8 */
    __I uint32_t PERIPHID7;          /**< Peripheral ID Register, offset: 0xFDC */
    __I uint32_t PERIPHID0;          /**< Peripheral ID Register, offset: 0xFE0 */
    __I uint32_t PERIPHID1;          /**< Peripheral ID Register, offset: 0xFE4 */
    __I uint32_t PERIPHID2;          /**< Peripheral ID Register, offset: 0xFE8 */
    __I uint32_t PERIPHID3;          /**< Peripheral ID Register, offset: 0xFEC */
    __I uint32_t COMPID[4];          /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
} ROM_Type, *ROM_MemMapPtr;

```

```

/*
----- */
-- ROM - Register accessor macros
----- */

```

```

/*!

* @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros

* @{

*/



/* ROM - Register accessors */

#define ROM_ENTRY_REG(base,index)      ((base)->ENTRY[index])

#define ROM_ENTRY_COUNT           3

#define ROM_TABLEMARK_REG(base)     ((base)->TABLEMARK)

#define ROM_SYSACCESS_REG(base)    ((base)->SYSACCESS)

#define ROM_PERIPHID4_REG(base)   ((base)->PERIPHID4)

#define ROM_PERIPHID5_REG(base)   ((base)->PERIPHID5)

#define ROM_PERIPHID6_REG(base)   ((base)->PERIPHID6)

#define ROM_PERIPHID7_REG(base)   ((base)->PERIPHID7)

#define ROM_PERIPHID0_REG(base)   ((base)->PERIPHID0)

#define ROM_PERIPHID1_REG(base)   ((base)->PERIPHID1)

#define ROM_PERIPHID2_REG(base)   ((base)->PERIPHID2)

#define ROM_PERIPHID3_REG(base)   ((base)->PERIPHID3)

#define ROM_COMPID_REG(base,index) ((base)->COMPID[index])

#define ROM_COMPID_COUNT          4


/*!


* @}

*/ /* end of group ROM_Register_Accessor_Macros */

```

```
/* -----
```

```

-- ROM Register Masks
----- */
/*!
* @addtogroup ROM_Register_Masks ROM Register Masks
* @{
*/
/* ENTRY Bit Fields */

#define ROM_ENTRY_ENTRY_MASK          0xFFFFFFFFu
#define ROM_ENTRY_ENTRY_SHIFT         0
#define ROM_ENTRY_ENTRY_WIDTH         32
#define ROM_ENTRY_ENTRY(x)            (((uint32_t)((uint32_t)(x))<<ROM_ENTRY_ENTRY_SHIFT))&ROM_ENTRY_ENTRY_MASK

/* TABLEMARK Bit Fields */

#define ROM_TABLEMARK_MARK_MASK       0xFFFFFFFFu
#define ROM_TABLEMARK_MARK_SHIFT      0
#define ROM_TABLEMARK_MARK_WIDTH      32
#define ROM_TABLEMARK_MARK(x)         (((uint32_t)((uint32_t)(x))<<ROM_TABLEMARK_MARK_SHIFT))&ROM_TABLEMARK_MARK_MASK

/* SYSACCESS Bit Fields */

#define ROM_SYSACCESS_SYSACCESS_MASK  0xFFFFFFFFu
#define ROM_SYSACCESS_SYSACCESS_SHIFT 0
#define ROM_SYSACCESS_SYSACCESS_WIDTH 32
#define ROM_SYSACCESS_SYSACCESS(x)    (((uint32_t)((uint32_t)(x))<<ROM_SYSACCESS_SYSACCESS_SHIFT))&ROM_SYSACCESS_SYSACCESS_MASK

/* PERIPHID4 Bit Fields */

#define ROM_PERIPHID4_PERIPHID_MASK   0xFFFFFFFFu
#define ROM_PERIPHID4_PERIPHID_SHIFT  0

```

```

#define ROM_PERIPHID4_PERIPHID_WIDTH      32
#define ROM_PERIPHID4_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID4_PERIPHID_SHIFT))&ROM_PERIPHID4_PERIPHID_MASK)

/* PERIPHID5 Bit Fields */

#define ROM_PERIPHID5_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID5_PERIPHID_SHIFT       0
#define ROM_PERIPHID5_PERIPHID_WIDTH       32
#define ROM_PERIPHID5_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID5_PERIPHID_SHIFT))&ROM_PERIPHID5_PERIPHID_MASK)

/* PERIPHID6 Bit Fields */

#define ROM_PERIPHID6_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID6_PERIPHID_SHIFT       0
#define ROM_PERIPHID6_PERIPHID_WIDTH       32
#define ROM_PERIPHID6_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID6_PERIPHID_SHIFT))&ROM_PERIPHID6_PERIPHID_MASK)

/* PERIPHID7 Bit Fields */

#define ROM_PERIPHID7_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID7_PERIPHID_SHIFT       0
#define ROM_PERIPHID7_PERIPHID_WIDTH       32
#define ROM_PERIPHID7_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID7_PERIPHID_SHIFT))&ROM_PERIPHID7_PERIPHID_MASK)

/* PERIPHID0 Bit Fields */

#define ROM_PERIPHIDO_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHIDO_PERIPHID_SHIFT       0
#define ROM_PERIPHIDO_PERIPHID_WIDTH       32
#define ROM_PERIPHIDO_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHIDO_PERIPHID_SHIFT))&ROM_PERIPHIDO_PERIPHID_MASK)

/* PERIPHID1 Bit Fields */

#define ROM_PERIPHID1_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID1_PERIPHID_SHIFT       0

```

```

#define ROM_PERIPHID1_PERIPHID_WIDTH      32
#define ROM_PERIPHID1_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID1_PERIPHID_SHIFT))&ROM_PERIPHID1_PERIPHID_MASK)

/* PERIPHID2 Bit Fields */

#define ROM_PERIPHID2_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID2_PERIPHID_SHIFT       0
#define ROM_PERIPHID2_PERIPHID_WIDTH       32
#define ROM_PERIPHID2_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID2_PERIPHID_SHIFT))&ROM_PERIPHID2_PERIPHID_MASK)

/* PERIPHID3 Bit Fields */

#define ROM_PERIPHID3_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID3_PERIPHID_SHIFT       0
#define ROM_PERIPHID3_PERIPHID_WIDTH       32
#define ROM_PERIPHID3_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID3_PERIPHID_SHIFT))&ROM_PERIPHID3_PERIPHID_MASK)

/* COMPID Bit Fields */

#define ROM_COMPID_COMPID_MASK             0xFFFFFFFFu
#define ROM_COMPID_COMPID_SHIFT            0
#define ROM_COMPID_COMPID_WIDTH           32
#define ROM_COMPID_COMPID(x)
(((uint32_t)((uint32_t)(x)<<ROM_COMPID_COMPID_SHIFT))&ROM_COMPID_COMPID_MASK)

/*!
 * @}
 */

/* /* end of group ROM_Register_Masks */

```

```

/* ROM - Peripheral instance base addresses */
/** Peripheral ROM base address */
#define ROM_BASE                      (0xF0002000u)

```

```

/** Peripheral ROM base pointer */

#define ROM           ((ROM_Type *)ROM_BASE)

#define ROM_BASE_PTR      (ROM)

/** Array initializer of ROM peripheral base addresses */

#define ROM_BASE_ADDRS    { ROM_BASE }

/** Array initializer of ROM peripheral base pointers */

#define ROM_BASE_PTRS     { ROM }

/*
-----*
-- ROM - Register accessor macros
----- */
/*!
* @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros
* @{
*/
/* ROM - Register instance definitions */

/* ROM */

#define ROM_ENTRY0        ROM_ENTRY_REG(ROM,0)
#define ROM_ENTRY1        ROM_ENTRY_REG(ROM,1)
#define ROM_ENTRY2        ROM_ENTRY_REG(ROM,2)
#define ROM_TABLEMARK     ROM_TABLEMARK_REG(ROM)
#define ROM_SYSACCESS     ROM_SYSACCESS_REG(ROM)
#define ROM_PERIPHID4    ROM_PERIPHID4_REG(ROM)
#define ROM_PERIPHID5    ROM_PERIPHID5_REG(ROM)
#define ROM_PERIPHID6    ROM_PERIPHID6_REG(ROM)
#define ROM_PERIPHID7    ROM_PERIPHID7_REG(ROM)

```

```
#define ROM_PERIPHID0      ROM_PERIPHID0_REG(ROM)
#define ROM_PERIPHID1      ROM_PERIPHID1_REG(ROM)
#define ROM_PERIPHID2      ROM_PERIPHID2_REG(ROM)
#define ROM_PERIPHID3      ROM_PERIPHID3_REG(ROM)
#define ROM_COMPID0          ROM_COMPID_REG(ROM,0)
#define ROM_COMPID1          ROM_COMPID_REG(ROM,1)
#define ROM_COMPID2          ROM_COMPID_REG(ROM,2)
#define ROM_COMPID3          ROM_COMPID_REG(ROM,3)
```

```
/* ROM - Register array accessors */
```

```
#define ROM_ENTRY(index)    ROM_ENTRY_REG(ROM,index)
#define ROM_COMPID(index)    ROM_COMPID_REG(ROM,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group ROM_Register_Accessor_Macros */
```

```
/*!
```

```
* @}
```

```
/* /* end of group ROM_Peripheral_Access_Layer */
```

```
/* -----
```

```
-- RTC Peripheral Access Layer
```

```
----- */
```

```
/*!
```

```
* @addtogroup RTC_Peripheral_Access_Layer RTC Peripheral Access Layer
```

```

* @@
*/

```

/** RTC - Register Layout Typedef */

```

typedef struct {
    __IO uint32_t TSR;           /**< RTC Time Seconds Register, offset: 0x0 */
    __IO uint32_t TPR;           /**< RTC Time Prescaler Register, offset: 0x4 */
    __IO uint32_t TAR;           /**< RTC Time Alarm Register, offset: 0x8 */
    __IO uint32_t TCR;           /**< RTC Time Compensation Register, offset: 0xC */
    __IO uint32_t CR;            /**< RTC Control Register, offset: 0x10 */
    __IO uint32_t SR;            /**< RTC Status Register, offset: 0x14 */
    __IO uint32_t LR;            /**< RTC Lock Register, offset: 0x18 */
    __IO uint32_t IER;           /**< RTC Interrupt Enable Register, offset: 0x1C */
}
RTC_Type, *RTC_MemMapPtr;

```

```

/*
-----  

-- RTC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros  

* @@
*/

```

```

/* RTC - Register accessors */  

#define RTC_TSREG(base)          ((base)->TSR)  

#define RTC_TPREG(base)          ((base)->TPR)  

#define RTC_TARREG(base)         ((base)->TAR)

```

```
#define RTC_TCR_REG(base) ((base)->TCR)
#define RTC_CR_REG(base) ((base)->CR)
#define RTC_SR_REG(base) ((base)->SR)
#define RTC_LR_REG(base) ((base)->LR)
#define RTC_IER_REG(base) ((base)->IER)
```

```
/*!
 * @}
 */
/* /* end of group RTC_Register_Accessor_Macros */
```

```
/*
-----  
-- RTC Register Masks  
----- */
```

```
/*!
 * @addtogroup RTC_Register_Masks RTC Register Masks
 * @{
 */

```

```
/* TSR Bit Fields */

#define RTC_TSRR_MASK 0xFFFFFFFFu
#define RTC_TSRR_SHIFT 0
#define RTC_TSRR_WIDTH 32

#define RTC_TSRR(x) (((uint32_t)((uint32_t)(x))<<RTC_TSRR_SHIFT))&RTC_TSRR_MASK

/* TPR Bit Fields */

#define RTC_TPR_TPR_MASK 0xFFFFu
#define RTC_TPR_TPR_SHIFT 0
```

```

#define RTC_TPR_TPR_WIDTH          16
#define RTC_TPR_TPR(x) (((uint32_t)((uint32_t)(x))<<RTC_TPR_TPR_SHIFT))&RTC_TPR_TPR_MASK)
/* TAR Bit Fields */

#define RTC_TAR_TAR_MASK           0xFFFFFFFFu
#define RTC_TAR_TAR_SHIFT          0
#define RTC_TAR_TAR_WIDTH          32
#define RTC_TAR_TAR(x) (((uint32_t)((uint32_t)(x))<<RTC_TAR_TAR_SHIFT))&RTC_TAR_TAR_MASK)

/* TCR Bit Fields */

#define RTC_TCR_TCR_MASK           0xFFu
#define RTC_TCR_TCR_SHIFT          0
#define RTC_TCR_TCR_WIDTH          8
#define RTC_TCR_TCR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCR_SHIFT))&RTC_TCR_TCR_MASK)

#define RTC_TCR_CIR_MASK           0xFF00u
#define RTC_TCR_CIR_SHIFT          8
#define RTC_TCR_CIR_WIDTH          8
#define RTC_TCR_CIR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIR_SHIFT))&RTC_TCR_CIR_MASK)

#define RTC_TCR_TCV_MASK           0xFF0000u
#define RTC_TCR_TCV_SHIFT          16
#define RTC_TCR_TCV_WIDTH          8
#define RTC_TCR_TCV(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCV_SHIFT))&RTC_TCR_TCV_MASK)

#define RTC_TCR_CIC_MASK           0xFF000000u
#define RTC_TCR_CIC_SHIFT          24
#define RTC_TCR_CIC_WIDTH          8
#define RTC_TCR_CIC(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIC_SHIFT))&RTC_TCR_CIC_MASK)

/* CR Bit Fields */

```

```

#define RTC_CR_SWR_MASK           0x1u
#define RTC_CR_SWR_SHIFT          0
#define RTC_CR_SWR_WIDTH          1
#define RTC_CR_SWR(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_SWR_SHIFT))&RTC_CR_SWR_MASK)
#define RTC_CR_WPE_MASK            0x2u
#define RTC_CR_WPE_SHIFT           1
#define RTC_CR_WPE_WIDTH           1
#define RTC_CR_WPE(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_WPE_SHIFT))&RTC_CR_WPE_MASK)
#define RTC_CR_SUP_MASK             0x4u
#define RTC_CR_SUP_SHIFT            2
#define RTC_CR_SUP_WIDTH            1
#define RTC_CR_SUP(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_SUP_SHIFT))&RTC_CR_SUP_MASK)
#define RTC_CR_UM_MASK              0x8u
#define RTC_CR_UM_SHIFT              3
#define RTC_CR_UM_WIDTH              1
#define RTC_CR_UM(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_UM_SHIFT))&RTC_CR_UM_MASK)
#define RTC_CR_OSCE_MASK             0x100u
#define RTC_CR_OSCE_SHIFT             8
#define RTC_CR_OSCE_WIDTH             1
#define RTC_CR_OSCE(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_OSCE_SHIFT))&RTC_CR_OSCE_MASK)
#define RTC_CR_CLKO_MASK             0x200u
#define RTC_CR_CLKO_SHIFT             9
#define RTC_CR_CLKO_WIDTH             1
#define RTC_CR_CLKO(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_CLKO_SHIFT))&RTC_CR_CLKO_MASK)
#define RTC_CR_SC16P_MASK             0x400u

```

```

#define RTC_CR_SC16P_SHIFT          10
#define RTC_CR_SC16P_WIDTH          1
#define RTC_CR_SC16P(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_SC16P_SHIFT))&RTC_CR_SC16P_MASK)
#define RTC_CR_SC8P_MASK            0x800u
#define RTC_CR_SC8P_SHIFT           11
#define RTC_CR_SC8P_WIDTH           1
#define RTC_CR_SC8P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC8P_SHIFT))&RTC_CR_SC8P_MASK)
#define RTC_CR_SC4P_MASK            0x1000u
#define RTC_CR_SC4P_SHIFT           12
#define RTC_CR_SC4P_WIDTH           1
#define RTC_CR_SC4P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC4P_SHIFT))&RTC_CR_SC4P_MASK)
#define RTC_CR_SC2P_MASK            0x2000u
#define RTC_CR_SC2P_SHIFT           13
#define RTC_CR_SC2P_WIDTH           1
#define RTC_CR_SC2P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC2P_SHIFT))&RTC_CR_SC2P_MASK)

/* SR Bit Fields */

#define RTC_SR_TIF_MASK             0x1u
#define RTC_SR_TIF_SHIFT             0
#define RTC_SR_TIF_WIDTH             1
#define RTC_SR_TIF(x)                (((uint32_t)((uint32_t)(x))<<RTC_SR_TIF_SHIFT))&RTC_SR_TIF_MASK)
#define RTC_SR_TOF_MASK              0x2u
#define RTC_SR_TOF_SHIFT              1
#define RTC_SR_TOF_WIDTH              1
#define RTC_SR_TOF(x)                 (((uint32_t)((uint32_t)(x))<<RTC_SR_TOF_SHIFT))&RTC_SR_TOF_MASK)
#define RTC_SR_TAF_MASK              0x4u

```

```

#define RTC_SR_TAF_SHIFT          2
#define RTC_SR_TAF_WIDTH          1
#define RTC_SR_TAF(x)
(((uint32_t)((uint32_t)(x)<<RTC_SR_TAF_SHIFT))&RTC_SR_TAF_MASK)

#define RTC_SR_TCE_MASK           0x10u
#define RTC_SR_TCE_SHIFT          4
#define RTC_SR_TCE_WIDTH          1
#define RTC_SR_TCE(x)
(((uint32_t)((uint32_t)(x)<<RTC_SR_TCE_SHIFT))&RTC_SR_TCE_MASK)

/* LR Bit Fields */

#define RTC_LR_TCL_MASK           0x8u
#define RTC_LR_TCL_SHIFT          3
#define RTC_LR_TCL_WIDTH          1
#define RTC_LR_TCL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_TCL_SHIFT))&RTC_LR_TCL_MASK)

#define RTC_LR_CRL_MASK           0x10u
#define RTC_LR_CRL_SHIFT          4
#define RTC_LR_CRL_WIDTH          1
#define RTC_LR_CRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_CRL_SHIFT))&RTC_LR_CRL_MASK)

#define RTC_LR_SRL_MASK           0x20u
#define RTC_LR_SRL_SHIFT          5
#define RTC_LR_SRL_WIDTH          1
#define RTC_LR_SRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_SRL_SHIFT))&RTC_LR_SRL_MASK)

#define RTC_LR_LRL_MASK           0x40u
#define RTC_LR_LRL_SHIFT          6
#define RTC_LR_LRL_WIDTH          1
#define RTC_LR_LRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_LRL_SHIFT))&RTC_LR_LRL_MASK)

/* IER Bit Fields */

```

```

#define RTC_IER_TIIE_MASK           0x1u
#define RTC_IER_TIIE_SHIFT          0
#define RTC_IER_TIIE_WIDTH          1
#define RTC_IER_TIIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TIIE_SHIFT))&RTC_IER_TIIE_MASK

#define RTC_IER_TOIE_MASK           0x2u
#define RTC_IER_TOIE_SHIFT          1
#define RTC_IER_TOIE_WIDTH          1
#define RTC_IER_TOIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TOIE_SHIFT))&RTC_IER_TOIE_MASK

#define RTC_IER_TAIE_MASK           0x4u
#define RTC_IER_TAIE_SHIFT          2
#define RTC_IER_TAIE_WIDTH          1
#define RTC_IER_TAIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TAIE_SHIFT))&RTC_IER_TAIE_MASK

#define RTC_IER_TSIE_MASK           0x10u
#define RTC_IER_TSIE_SHIFT          4
#define RTC_IER_TSIE_WIDTH          1
#define RTC_IER_TSIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TSIE_SHIFT))&RTC_IER_TSIE_MASK

#define RTC_IER_WPON_MASK           0x80u
#define RTC_IER_WPON_SHIFT          7
#define RTC_IER_WPON_WIDTH          1
#define RTC_IER_WPON(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_WPON_SHIFT))&RTC_IER_WPON_MASK

/*
 * @}
 */ /* end of group RTC_Register_Masks */

```

```

/* RTC - Peripheral instance base addresses */

/** Peripheral RTC base address */

#define RTC_BASE           (0x4003D000u)

/** Peripheral RTC base pointer */

#define RTC             ((RTC_Type *)RTC_BASE)

#define RTC_BASE_PTR      (RTC)

/** Array initializer of RTC peripheral base addresses */

#define RTC_BASE_ADDRS    { RTC_BASE }

/** Array initializer of RTC peripheral base pointers */

#define RTC_BASE_PTRS     { RTC }

/*
-----*
-- RTC - Register accessor macros
----- */
/*!
* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros
* @{
*/
/* RTC - Register instance definitions */

/* RTC */

#define RTC_TSR           RTC_TSREG(RTC)
#define RTC_TPR           RTC_TSPRREG(RTC)
#define RTC_TAR           RTC_TARREG(RTC)
#define RTC_TCR           RTC_TCRRREG(RTC)
#define RTC_CR            RTC_CRRREG(RTC)

```

```
#define RTC_SR           RTC_SR_REG(RTC)
#define RTC_LR           RTC_LR_REG(RTC)
#define RTC_IER           RTC_IER_REG(RTC)
```

```
/*!
 * @}
 */
/*/* end of group RTC_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/*/* end of group RTC_Peripheral_Access_Layer */
```

```
/*
-----  
-- SIM Peripheral Access Layer  
----- */
```

```
/*!
 * @addtogroup SIM_Peripheral_Access_Layer SIM Peripheral Access Layer
 * @{
 */

```

```
/** SIM - Register Layout Typedef */
typedef struct {
    __IO uint32_t SOPT1;           /**< System Options Register 1, offset: 0x0 */
    __IO uint32_t SOPT1CFG;        /**< SOPT1 Configuration Register, offset: 0x4 */
    uint8_t RESERVED_0[4092];
    __IO uint32_t SOPT2;           /**< System Options Register 2, offset: 0x1004 */
}
```

```

    uint8_t RESERVED_1[4];

    __IO uint32_t SOPT4;           /**< System Options Register 4, offset: 0x100C */
    __IO uint32_t SOPT5;           /**< System Options Register 5, offset: 0x1010 */
    uint8_t RESERVED_2[4];
    __IO uint32_t SOPT7;           /**< System Options Register 7, offset: 0x1018 */
    uint8_t RESERVED_3[8];
    __I uint32_t SDID;            /**< System Device Identification Register, offset: 0x1024 */
    uint8_t RESERVED_4[12];
    __IO uint32_t SCGC4;           /**< System Clock Gating Control Register 4, offset: 0x1034 */
    __IO uint32_t SCGC5;           /**< System Clock Gating Control Register 5, offset: 0x1038 */
    __IO uint32_t SCGC6;           /**< System Clock Gating Control Register 6, offset: 0x103C */
    __IO uint32_t SCGC7;           /**< System Clock Gating Control Register 7, offset: 0x1040 */
    __IO uint32_t CLKDIV1;          /**< System Clock Divider Register 1, offset: 0x1044 */
    uint8_t RESERVED_5[4];
    __IO uint32_t FCFG1;           /**< Flash Configuration Register 1, offset: 0x104C */
    __I uint32_t FCFG2;           /**< Flash Configuration Register 2, offset: 0x1050 */
    uint8_t RESERVED_6[4];
    __I uint32_t UIDMH;            /**< Unique Identification Register Mid-High, offset: 0x1058 */
    __I uint32_t UIDML;            /**< Unique Identification Register Mid Low, offset: 0x105C */
    __I uint32_t UIDL;             /**< Unique Identification Register Low, offset: 0x1060 */
    uint8_t RESERVED_7[156];
    __IO uint32_t COPC;            /**< COP Control Register, offset: 0x1100 */
    __O uint32_t SRVCOP;           /**< Service COP Register, offset: 0x1104 */

} SIM_Type, *SIM_MemMapPtr;

```

```

/*
-----*
-- SIM - Register accessor macros
----- */

```

```

/*!

* @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros

* @{

*/



/* SIM - Register accessors */

#define SIM_SOPT1_REG(base)          ((base)->SOPT1)
#define SIM_SOPT1CFG_REG(base)       ((base)->SOPT1CFG)
#define SIM_SOPT2_REG(base)          ((base)->SOPT2)
#define SIM_SOPT4_REG(base)          ((base)->SOPT4)
#define SIM_SOPT5_REG(base)          ((base)->SOPT5)
#define SIM_SOPT7_REG(base)          ((base)->SOPT7)
#define SIM_SDID_REG(base)           ((base)->SDID)
#define SIM_SCGC4_REG(base)          ((base)->SCGC4)
#define SIM_SCGC5_REG(base)          ((base)->SCGC5)
#define SIM_SCGC6_REG(base)          ((base)->SCGC6)
#define SIM_SCGC7_REG(base)          ((base)->SCGC7)
#define SIM_CLKDIV1_REG(base)         ((base)->CLKDIV1)
#define SIM_FCFG1_REG(base)          ((base)->FCFG1)
#define SIM_FCFG2_REG(base)          ((base)->FCFG2)
#define SIM_UIDMH_REG(base)          ((base)->UIDMH)
#define SIM_UIDML_REG(base)          ((base)->UIDML)
#define SIM_UIDL_REG(base)           ((base)->UIDL)
#define SIM_COPC_REG(base)           ((base)->COPC)
#define SIM_SRVCOP_REG(base)         ((base)->SRVCOP)

!



* @}

```

```
/* /* end of group SIM_Register_Accessor_Macros */
```

```
/* -----  
-- SIM Register Masks  
----- */
```

```
/*!  
 * @addtogroup SIM_Register_Masks SIM Register Masks  
 * @{  
 */
```

```
/* SOPT1 Bit Fields */  
  
#define SIM_SOPT1_OSC32KSEL_MASK      0xC0000u  
#define SIM_SOPT1_OSC32KSEL_SHIFT     18  
#define SIM_SOPT1_OSC32KSEL_WIDTH     2  
  
#define SIM_SOPT1_OSC32KSEL(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_OSC32KSEL_SHIFT))&SIM_SOPT1_OSC32KSEL_MASK)  
  
#define SIM_SOPT1_USBVSTBY_MASK       0x20000000u  
#define SIM_SOPT1_USBVSTBY_SHIFT     29  
#define SIM_SOPT1_USBVSTBY_WIDTH     1  
  
#define SIM_SOPT1_USBVSTBY(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBVSTBY_SHIFT))&SIM_SOPT1_USBVSTBY_MASK)  
  
#define SIM_SOPT1_USBSSTBY_MASK      0x40000000u  
#define SIM_SOPT1_USBSSTBY_SHIFT     30  
#define SIM_SOPT1_USBSSTBY_WIDTH     1  
  
#define SIM_SOPT1_USBSSTBY(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBSSTBY_SHIFT))&SIM_SOPT1_USBSSTBY_MASK)  
  
#define SIM_SOPT1_USBREGEN_MASK      0x80000000u  
#define SIM_SOPT1_USBREGEN_SHIFT     31
```

```

#define SIM_SOPT1_USBREGEN_WIDTH          1

#define SIM_SOPT1_USBREGEN(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT1_USBREGEN_SHIFT))&SIM_SOPT1_USBREGEN_MASK)

/* SOPT1CFG Bit Fields */

#define SIM_SOPT1CFG_URWE_MASK           0x1000000u
#define SIM_SOPT1CFG_URWE_SHIFT         24
#define SIM_SOPT1CFG_URWE_WIDTH         1

#define SIM_SOPT1CFG_URWE(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT1CFG_URWE_SHIFT))&SIM_SOPT1CFG_URWE_MASK)

#define SIM_SOPT1CFG_UVSWE_MASK          0x2000000u
#define SIM_SOPT1CFG_UVSWE_SHIFT        25
#define SIM_SOPT1CFG_UVSWE_WIDTH         1

#define SIM_SOPT1CFG_UVSWE(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT1CFG_UVSWE_SHIFT))&SIM_SOPT1CFG_UVSWE_MASK)

#define SIM_SOPT1CFG_USSWE_MASK          0x4000000u
#define SIM_SOPT1CFG_USSWE_SHIFT        26
#define SIM_SOPT1CFG_USSWE_WIDTH         1

#define SIM_SOPT1CFG_USSWE(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT1CFG_USSWE_SHIFT))&SIM_SOPT1CFG_USSWE_MASK)

/* SOPT2 Bit Fields */

#define SIM_SOPT2_RTCCLKOUTSEL_MASK      0x10u
#define SIM_SOPT2_RTCCLKOUTSEL_SHIFT    4
#define SIM_SOPT2_RTCCLKOUTSEL_WIDTH    1

#define SIM_SOPT2_RTCCLKOUTSEL(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT2_RTCCLKOUTSEL_SHIFT))&SIM_SOPT2_RTCCLKOUTSEL_MASK)

#define SIM_SOPT2_CLKOUTSEL_MASK         0xE0u
#define SIM_SOPT2_CLKOUTSEL_SHIFT       5
#define SIM_SOPT2_CLKOUTSEL_WIDTH       3

#define SIM_SOPT2_CLKOUTSEL(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT2_CLKOUTSEL_SHIFT))&SIM_SOPT2_CLKOUTSEL_MASK)

#define SIM_SOPT2_PLLFLLSEL_MASK        0x10000u

```

```

#define SIM_SOPT2_PLLFLLSEL_SHIFT      16
#define SIM_SOPT2_PLLFLLSEL_WIDTH      1
#define SIM_SOPT2_PLLFLLSEL(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_PLLFLLSEL_SHIFT))&SIM_SOPT2_PLLFLLSEL_MASK)
#define SIM_SOPT2_USBSRC_MASK          0x40000u
#define SIM_SOPT2_USBSRC_SHIFT         18
#define SIM_SOPT2_USBSRC_WIDTH         1
#define SIM_SOPT2_USBSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_USBSRC_SHIFT))&SIM_SOPT2_USBSRC_MASK)
#define SIM_SOPT2_TPMSRC_MASK          0x3000000u
#define SIM_SOPT2_TPMSRC_SHIFT         24
#define SIM_SOPT2_TPMSRC_WIDTH         2
#define SIM_SOPT2_TPMSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_TPMSRC_SHIFT))&SIM_SOPT2_TPMSRC_MASK)
#define SIM_SOPT2_UART0SRC_MASK         0xC000000u
#define SIM_SOPT2_UART0SRC_SHIFT        26
#define SIM_SOPT2_UART0SRC_WIDTH        2
#define SIM_SOPT2_UART0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_UART0SRC_SHIFT))&SIM_SOPT2_UART0SRC_MASK)

/* SOPT4 Bit Fields */

#define SIM_SOPT4 TPM1CH0SRC_MASK       0x40000u
#define SIM_SOPT4 TPM1CH0SRC_SHIFT      18
#define SIM_SOPT4 TPM1CH0SRC_WIDTH      1
#define SIM_SOPT4 TPM1CH0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM1CH0SRC_SHIFT))&SIM_SOPT4 TPM1CH0SRC_MASK)
#define SIM_SOPT4 TPM2CH0SRC_MASK       0x100000u
#define SIM_SOPT4 TPM2CH0SRC_SHIFT      20
#define SIM_SOPT4 TPM2CH0SRC_WIDTH      1
#define SIM_SOPT4 TPM2CH0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM2CH0SRC_SHIFT))&SIM_SOPT4 TPM2CH0SRC_MASK)
#define SIM_SOPT4 TPM0CLKSEL_MASK       0x1000000u

```

```

#define SIM_SOPT4 TPM0CLKSEL_SHIFT      24
#define SIM_SOPT4 TPM0CLKSEL_WIDTH      1
#define SIM_SOPT4 TPM0CLKSEL(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT4 TPM0CLKSEL_SHIFT))&SIM_SOPT4 TPM0CLKSEL_MASK)

#define SIM_SOPT4 TPM1CLKSEL_MASK      0x2000000u
#define SIM_SOPT4 TPM1CLKSEL_SHIFT     25
#define SIM_SOPT4 TPM1CLKSEL_WIDTH     1
#define SIM_SOPT4 TPM1CLKSEL(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT4 TPM1CLKSEL_SHIFT))&SIM_SOPT4 TPM1CLKSEL_MASK)

#define SIM_SOPT4 TPM2CLKSEL_MASK      0x4000000u
#define SIM_SOPT4 TPM2CLKSEL_SHIFT     26
#define SIM_SOPT4 TPM2CLKSEL_WIDTH     1
#define SIM_SOPT4 TPM2CLKSEL(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT4 TPM2CLKSEL_SHIFT))&SIM_SOPT4 TPM2CLKSEL_MASK)

/* SOPT5 Bit Fields */

#define SIM_SOPT5 UART0TXSRC_MASK      0x3u
#define SIM_SOPT5 UART0TXSRC_SHIFT     0
#define SIM_SOPT5 UART0TXSRC_WIDTH     2
#define SIM_SOPT5 UART0TXSRC(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT5 UART0TXSRC_SHIFT))&SIM_SOPT5 UART0TXSRC_MASK)

#define SIM_SOPT5 UART0RXSRC_MASK      0x4u
#define SIM_SOPT5 UART0RXSRC_SHIFT     2
#define SIM_SOPT5 UART0RXSRC_WIDTH     1
#define SIM_SOPT5 UART0RXSRC(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT5 UART0RXSRC_SHIFT))&SIM_SOPT5 UART0RXSRC_MASK)

#define SIM_SOPT5 UART1TXSRC_MASK      0x30u
#define SIM_SOPT5 UART1TXSRC_SHIFT     4
#define SIM_SOPT5 UART1TXSRC_WIDTH     2
#define SIM_SOPT5 UART1TXSRC(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT5 UART1TXSRC_SHIFT))&SIM_SOPT5 UART1TXSRC_MASK

#define SIM_SOPT5 UART1RXSRC_MASK      0x40u

```

```

#define SIM_SOPT5_UART1RXSRC_SHIFT      6
#define SIM_SOPT5_UART1RXSRC_WIDTH      1
#define SIM_SOPT5_UART1RXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART1RXSRC_SHIFT))&SIM_SOPT5_UART1RXSRC_MASK)

#define SIM_SOPT5_UART0ODE_MASK         0x10000u
#define SIM_SOPT5_UART0ODE_SHIFT        16
#define SIM_SOPT5_UART0ODE_WIDTH        1
#define SIM_SOPT5_UART0ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART0ODE_SHIFT))&SIM_SOPT5_UART0ODE_MASK)

#define SIM_SOPT5_UART1ODE_MASK         0x20000u
#define SIM_SOPT5_UART1ODE_SHIFT        17
#define SIM_SOPT5_UART1ODE_WIDTH        1
#define SIM_SOPT5_UART1ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART1ODE_SHIFT))&SIM_SOPT5_UART1ODE_MASK)

#define SIM_SOPT5_UART2ODE_MASK         0x40000u
#define SIM_SOPT5_UART2ODE_SHIFT        18
#define SIM_SOPT5_UART2ODE_WIDTH        1
#define SIM_SOPT5_UART2ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART2ODE_SHIFT))&SIM_SOPT5_UART2ODE_MASK)

/* SOPT7 Bit Fields */

#define SIM_SOPT7_ADC0TRGSEL_MASK       0xFu
#define SIM_SOPT7_ADC0TRGSEL_SHIFT      0
#define SIM_SOPT7_ADC0TRGSEL_WIDTH      4
#define SIM_SOPT7_ADC0TRGSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADC0TRGSEL_SHIFT))&SIM_SOPT7_ADC0TRGSEL_MASK)

#define SIM_SOPT7_ADCOPRETRGSEL_MASK    0x10u
#define SIM_SOPT7_ADCOPRETRGSEL_SHIFT   4
#define SIM_SOPT7_ADCOPRETRGSEL_WIDTH   1
#define SIM_SOPT7_ADCOPRETRGSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADCOPRETRGSEL_SHIFT))&SIM_SOPT7_ADCOPRETRGSEL_MASK)

```

```

#define SIM_SOPT7_ADCOALTTRGEN_MASK      0x80u
#define SIM_SOPT7_ADCOALTTRGEN_SHIFT     7
#define SIM_SOPT7_ADCOALTTRGEN_WIDTH     1
#define SIM_SOPT7_ADCOALTTRGEN(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADCOALTTRGEN_SHIFT))&SIM_SOPT7_ADCOALTTRGEN_MASK)

/* SDID Bit Fields */

#define SIM_SDID_PINID_MASK            0xFu
#define SIM_SDID_PINID_SHIFT           0
#define SIM_SDID_PINID_WIDTH          4
#define SIM_SDID_PINID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_PINID_SHIFT))&SIM_SDID_PINID_MASK)

#define SIM_SDID_DIEID_MASK           0xF80u
#define SIM_SDID_DIEID_SHIFT          7
#define SIM_SDID_DIEID_WIDTH          5
#define SIM_SDID_DIEID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_DIEID_SHIFT))&SIM_SDID_DIEID_MASK)

#define SIM_SDID_REVID_MASK           0xF000u
#define SIM_SDID_REVID_SHIFT          12
#define SIM_SDID_REVID_WIDTH          4
#define SIM_SDID_REVID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_REVID_SHIFT))&SIM_SDID_REVID_MASK)

#define SIM_SDID_SRAMSIZE_MASK         0xF0000u
#define SIM_SDID_SRAMSIZE_SHIFT        16
#define SIM_SDID_SRAMSIZE_WIDTH        4
#define SIM_SDID_SRAMSIZE(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_SRAMSIZE_SHIFT))&SIM_SDID_SRAMSIZE_MASK)

#define SIM_SDID_SERIESID_MASK         0xF00000u
#define SIM_SDID_SERIESID_SHIFT        20
#define SIM_SDID_SERIESID_WIDTH        4
#define SIM_SDID_SERIESID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_SERIESID_SHIFT))&SIM_SDID_SERIESID_MASK)

```

```

#define SIM_SDID_SUBFAMID_MASK          0xF000000u
#define SIM_SDID_SUBFAMID_SHIFT         24
#define SIM_SDID_SUBFAMID_WIDTH         4
#define SIM_SDID_SUBFAMID(x)           (((uint32_t)((uint32_t)(x))<<SIM_SDID_SUBFAMID_SHIFT))&SIM_SDID_SUBFAMID_MASK)

#define SIM_SDID_FAMID_MASK            0xF0000000u
#define SIM_SDID_FAMID_SHIFT           28
#define SIM_SDID_FAMID_WIDTH           4
#define SIM_SDID_FAMID(x)             (((uint32_t)((uint32_t)(x))<<SIM_SDID_FAMID_SHIFT))&SIM_SDID_FAMID_MASK)

/* SCGC4 Bit Fields */

#define SIM_SCGC4_I2C0_MASK            0x40u
#define SIM_SCGC4_I2C0_SHIFT           6
#define SIM_SCGC4_I2C0_WIDTH           1
#define SIM_SCGC4_I2C0(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_I2C0_SHIFT))&SIM_SCGC4_I2C0_MASK)

#define SIM_SCGC4_I2C1_MASK            0x80u
#define SIM_SCGC4_I2C1_SHIFT           7
#define SIM_SCGC4_I2C1_WIDTH           1
#define SIM_SCGC4_I2C1(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_I2C1_SHIFT))&SIM_SCGC4_I2C1_MASK)

#define SIM_SCGC4_UART0_MASK           0x400u
#define SIM_SCGC4_UART0_SHIFT          10
#define SIM_SCGC4_UART0_WIDTH          1
#define SIM_SCGC4_UART0(x)             (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_UART0_SHIFT))&SIM_SCGC4_UART0_MASK)

#define SIM_SCGC4_UART1_MASK           0x800u
#define SIM_SCGC4_UART1_SHIFT          11
#define SIM_SCGC4_UART1_WIDTH          1
#define SIM_SCGC4_UART1(x)             (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_UART1_SHIFT))&SIM_SCGC4_UART1_MASK)

```

```

#define SIM_SCGC4_UART2_MASK          0x1000u
#define SIM_SCGC4_UART2_SHIFT         12
#define SIM_SCGC4_UART2_WIDTH         1
#define SIM_SCGC4_UART2(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_UART2_SHIFT))&SIM_SCGC4_UART2_MASK)
#define SIM_SCGC4_USBOTG_MASK         0x40000u
#define SIM_SCGC4_USBOTG_SHIFT        18
#define SIM_SCGC4_USBOTG_WIDTH        1
#define SIM_SCGC4_USBOTG(x)           (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_USBOTG_SHIFT))&SIM_SCGC4_USBOTG_MASK)
#define SIM_SCGC4_CMP_MASK            0x80000u
#define SIM_SCGC4_CMP_SHIFT           19
#define SIM_SCGC4_CMP_WIDTH           1
#define SIM_SCGC4_CMP(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_CMP_SHIFT))&SIM_SCGC4_CMP_MASK)
#define SIM_SCGC4_SPI0_MASK           0x400000u
#define SIM_SCGC4_SPI0_SHIFT          22
#define SIM_SCGC4_SPI0_WIDTH          1
#define SIM_SCGC4_SPI0(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI0_SHIFT))&SIM_SCGC4_SPI0_MASK)
#define SIM_SCGC4_SPI1_MASK           0x800000u
#define SIM_SCGC4_SPI1_SHIFT          23
#define SIM_SCGC4_SPI1_WIDTH          1
#define SIM_SCGC4_SPI1(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI1_SHIFT))&SIM_SCGC4_SPI1_MASK)
/* SCGC5 Bit Fields */
#define SIM_SCGC5_LPTMR_MASK          0x1u
#define SIM_SCGC5_LPTMR_SHIFT         0
#define SIM_SCGC5_LPTMR_WIDTH         1
#define SIM_SCGC5_LPTMR(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_LPTMR_SHIFT))&SIM_SCGC5_LPTMR_MASK)

```

```

#define SIM_SCGC5_TSI_MASK          0x20u
#define SIM_SCGC5_TSI_SHIFT         5
#define SIM_SCGC5_TSI_WIDTH         1
#define SIM_SCGC5_TSI(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_TSI_SHIFT))&SIM_SCGC5_TSI_MASK)
#define SIM_SCGC5_PORTA_MASK        0x200u
#define SIM_SCGC5_PORTA_SHIFT       9
#define SIM_SCGC5_PORTA_WIDTH       1
#define SIM_SCGC5_PORTA(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTA_SHIFT))&SIM_SCGC5_PORTA_MASK)
#define SIM_SCGC5_PORTB_MASK        0x400u
#define SIM_SCGC5_PORTB_SHIFT       10
#define SIM_SCGC5_PORTB_WIDTH       1
#define SIM_SCGC5_PORTB(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTB_SHIFT))&SIM_SCGC5_PORTB_MASK)
#define SIM_SCGC5_PORTC_MASK        0x800u
#define SIM_SCGC5_PORTC_SHIFT       11
#define SIM_SCGC5_PORTC_WIDTH       1
#define SIM_SCGC5_PORTC(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTC_SHIFT))&SIM_SCGC5_PORTC_MASK)
#define SIM_SCGC5_PORTD_MASK        0x1000u
#define SIM_SCGC5_PORTD_SHIFT       12
#define SIM_SCGC5_PORTD_WIDTH       1
#define SIM_SCGC5_PORTD(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTD_SHIFT))&SIM_SCGC5_PORTD_MASK)
#define SIM_SCGC5 PORTE_MASK        0x2000u
#define SIM_SCGC5 PORTE_SHIFT       13
#define SIM_SCGC5 PORTE_WIDTH       1
#define SIM_SCGC5 PORTE(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5 PORTE_SHIFT))&SIM_SCGC5 PORTE_MASK)
/* SCGC6 Bit Fields */

```

```

#define SIM_SCGC6_FTF_MASK           0x1u
#define SIM_SCGC6_FTF_SHIFT          0
#define SIM_SCGC6_FTF_WIDTH          1
#define SIM_SCGC6_FTF(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_FTF_SHIFT))&SIM_SCGC6_FTF_MASK)

#define SIM_SCGC6_DMAMUX_MASK         0x2u
#define SIM_SCGC6_DMAMUX_SHIFT        1
#define SIM_SCGC6_DMAMUX_WIDTH        1
#define SIM_SCGC6_DMAMUX(x)           (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_DMAMUX_SHIFT))&SIM_SCGC6_DMAMUX_MASK)

#define SIM_SCGC6_PIT_MASK            0x800000u
#define SIM_SCGC6_PIT_SHIFT           23
#define SIM_SCGC6_PIT_WIDTH           1
#define SIM_SCGC6_PIT(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_PIT_SHIFT))&SIM_SCGC6_PIT_MASK)

#define SIM_SCGC6 TPM0_MASK           0x1000000u
#define SIM_SCGC6 TPM0_SHIFT           24
#define SIM_SCGC6 TPM0_WIDTH           1
#define SIM_SCGC6 TPM0(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM0_SHIFT))&SIM_SCGC6 TPM0_MASK)

#define SIM_SCGC6 TPM1_MASK            0x2000000u
#define SIM_SCGC6 TPM1_SHIFT           25
#define SIM_SCGC6 TPM1_WIDTH           1
#define SIM_SCGC6 TPM1(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM1_SHIFT))&SIM_SCGC6 TPM1_MASK)

#define SIM_SCGC6 TPM2_MASK            0x4000000u
#define SIM_SCGC6 TPM2_SHIFT           26
#define SIM_SCGC6 TPM2_WIDTH           1
#define SIM_SCGC6 TPM2(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM2_SHIFT))&SIM_SCGC6 TPM2_MASK)

#define SIM_SCGC6 ADC0_MASK            0x8000000u

```

```

#define SIM_SCGC6_ADC0_SHIFT          27
#define SIM_SCGC6_ADC0_WIDTH          1
#define SIM_SCGC6_ADC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_ADC0_SHIFT))&SIM_SCGC6_ADC0_MASK)
#define SIM_SCGC6_RTC_MASK            0x20000000u
#define SIM_SCGC6_RTC_SHIFT           29
#define SIM_SCGC6_RTC_WIDTH           1
#define SIM_SCGC6_RTC(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_RTC_SHIFT))&SIM_SCGC6_RTC_MASK)
#define SIM_SCGC6_DAC0_MASK            0x80000000u
#define SIM_SCGC6_DAC0_SHIFT           31
#define SIM_SCGC6_DAC0_WIDTH           1
#define SIM_SCGC6_DAC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_DAC0_SHIFT))&SIM_SCGC6_DAC0_MASK)

/* SCGC7 Bit Fields */
#define SIM_SCGC7_DMA_MASK             0x100u
#define SIM_SCGC7_DMA_SHIFT             8
#define SIM_SCGC7_DMA_WIDTH             1
#define SIM_SCGC7_DMA(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC7_DMA_SHIFT))&SIM_SCGC7_DMA_MASK)

/* CLKDIV1 Bit Fields */
#define SIM_CLKDIV1_OUTDIV4_MASK        0x70000u
#define SIM_CLKDIV1_OUTDIV4_SHIFT       16
#define SIM_CLKDIV1_OUTDIV4_WIDTH       3
#define SIM_CLKDIV1_OUTDIV4(x) (((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV4_SHIFT))&SIM_CLKDIV1_OUTDIV4_MASK)
#define SIM_CLKDIV1_OUTDIV1_MASK        0xF0000000u
#define SIM_CLKDIV1_OUTDIV1_SHIFT       28
#define SIM_CLKDIV1_OUTDIV1_WIDTH       4
#define SIM_CLKDIV1_OUTDIV1(x) (((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV1_SHIFT))&SIM_CLKDIV1_OUTDIV1_MASK)

```

```

/* FCFG1 Bit Fields */

#define SIM_FCFG1_FLASHDIS_MASK          0x1u
#define SIM_FCFG1_FLASHDIS_SHIFT        0
#define SIM_FCFG1_FLASHDIS_WIDTH        1
#define SIM_FCFG1_FLASHDIS(x)           (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDIS_SHIFT))&SIM_FCFG1_FLASHDIS_MASK)

#define SIM_FCFG1_FLASHDOZE_MASK         0x2u
#define SIM_FCFG1_FLASHDOZE_SHIFT       1
#define SIM_FCFG1_FLASHDOZE_WIDTH       1
#define SIM_FCFG1_FLASHDOZE(x)          (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDOZE_SHIFT))&SIM_FCFG1_FLASHDOZE_MASK)

#define SIM_FCFG1_PFSIZE_MASK           0xF000000u
#define SIM_FCFG1_PFSIZE_SHIFT          24
#define SIM_FCFG1_PFSIZE_WIDTH          4
#define SIM_FCFG1_PFSIZE(x)             (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_PFSIZE_SHIFT))&SIM_FCFG1_PFSIZE_MASK)

/* FCFG2 Bit Fields */

#define SIM_FCFG2_MAXADDR0_MASK          0x7F000000u
#define SIM_FCFG2_MAXADDR0_SHIFT         24
#define SIM_FCFG2_MAXADDR0_WIDTH         7
#define SIM_FCFG2_MAXADDR0(x)            (((uint32_t)((uint32_t)(x))<<SIM_FCFG2_MAXADDR0_SHIFT))&SIM_FCFG2_MAXADDR0_MASK)

/* UIDMH Bit Fields */

#define SIM_UIDMH_UID_MASK              0xFFFFu
#define SIM_UIDMH_UID_SHIFT             0
#define SIM_UIDMH_UID_WIDTH             16
#define SIM_UIDMH_UID(x)                (((uint32_t)((uint32_t)(x))<<SIM_UIDMH_UID_SHIFT))&SIM_UIDMH_UID_MASK)

/* UIDML Bit Fields */

#define SIM_UIDML_UID_MASK              0xFFFFFFFFu

```

```

#define SIM_UIDML_UID_SHIFT          0
#define SIM_UIDML_UID_WIDTH         32
#define SIM_UIDML_UID(x) (((uint32_t)((uint32_t)(x))<<SIM_UIDML_UID_SHIFT))&SIM_UIDML_UID_MASK)

/* UIDL Bit Fields */

#define SIM_UIDL_UID_MASK           0xFFFFFFFFu
#define SIM_UIDL_UID_SHIFT          0
#define SIM_UIDL_UID_WIDTH         32
#define SIM_UIDL_UID(x) (((uint32_t)((uint32_t)(x))<<SIM_UIDL_UID_SHIFT))&SIM_UIDL_UID_MASK)

/* COPC Bit Fields */

#define SIM_COPC_COPW_MASK          0x1u
#define SIM_COPC_COPW_SHIFT          0
#define SIM_COPC_COPW_WIDTH         1
#define SIM_COPC_COPW(x) (((uint32_t)((uint32_t)(x))<<SIM_COPC_COPW_SHIFT))&SIM_COPC_COPW_MASK)

#define SIM_COPC_COPCLKS_MASK        0x2u
#define SIM_COPC_COPCLKS_SHIFT       1
#define SIM_COPC_COPCLKS_WIDTH       1
#define SIM_COPC_COPCLKS(x) (((uint32_t)((uint32_t)(x))<<SIM_COPC_COPCLKS_SHIFT))&SIM_COPC_COPCLKS_MASK)

#define SIM_COPC_COPT_MASK           0xCu
#define SIM_COPC_COPT_SHIFT          2
#define SIM_COPC_COPT_WIDTH          2
#define SIM_COPC_COPT(x) (((uint32_t)((uint32_t)(x))<<SIM_COPC_COPT_SHIFT))&SIM_COPC_COPT_MASK)

/* SRVCOP Bit Fields */

#define SIM_SRVCOP_SRVCOP_MASK        0xFFu
#define SIM_SRVCOP_SRVCOP_SHIFT       0
#define SIM_SRVCOP_SRVCOP_WIDTH       8

```

```

#define SIM_SRVCOP_SRVCOP(x)
(((uint32_t)((uint32_t)(x))<<SIM_SRVCOP_SRVCOP_SHIFT))&SIM_SRVCOP_SRVCOP_MASK)

/*
 * @}
 */

/* SIM - Peripheral instance base addresses */

/** Peripheral SIM base address */

#define SIM_BASE          (0x40047000u)

/** Peripheral SIM base pointer */

#define SIM             ((SIM_Type *)SIM_BASE)

#define SIM_BASE_PTR      (SIM)

/** Array initializer of SIM peripheral base addresses */

#define SIM_BASE_ADDRS     { SIM_BASE }

/** Array initializer of SIM peripheral base pointers */

#define SIM_BASE_PTRS      { SIM }

/*
-----  

-- SIM - Register accessor macros  

----- */

```

```

/*
 * @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros
 * @{
 */


```

```

/* SIM - Register instance definitions */

/* SIM */

#define SIM_SOPT1           SIM_SOPT1_REG(SIM)
#define SIM_SOPT1CFG        SIM_SOPT1CFG_REG(SIM)
#define SIM_SOPT2           SIM_SOPT2_REG(SIM)
#define SIM_SOPT4           SIM_SOPT4_REG(SIM)
#define SIM_SOPT5           SIM_SOPT5_REG(SIM)
#define SIM_SOPT7           SIM_SOPT7_REG(SIM)
#define SIM_SDID            SIM_SDID_REG(SIM)
#define SIM_SCGC4           SIM_SCGC4_REG(SIM)
#define SIM_SCGC5           SIM_SCGC5_REG(SIM)
#define SIM_SCGC6           SIM_SCGC6_REG(SIM)
#define SIM_SCGC7           SIM_SCGC7_REG(SIM)
#define SIM_CLKDIV1         SIM_CLKDIV1_REG(SIM)
#define SIM_FCFG1            SIM_FCFG1_REG(SIM)
#define SIM_FCFG2            SIM_FCFG2_REG(SIM)
#define SIM_UIDMH           SIM_UIDMH_REG(SIM)
#define SIM_UIDML           SIM_UIDML_REG(SIM)
#define SIM_UIDL             SIM_UIDL_REG(SIM)
#define SIM_COPC             SIM_COPC_REG(SIM)
#define SIM_SRVCOP           SIM_SRVCOP_REG(SIM)

```

```

/*
 * @}
 */
 /* end of group SIM_Register_Accessor_Macros */

```

```

/*
 * @}

```

```
*/ /* end of group SIM_Peripheral_Access_Layer */
```

```
/* -----  
-- SMC Peripheral Access Layer  
----- */
```

```
/*!  
* @addtogroup SMC_Peripheral_Access_Layer SMC Peripheral Access Layer  
* @{  
*/
```

```
/** SMC - Register Layout Typedef */  
  
typedef struct {  
  
    __IO uint8_t PMPROT;           /**< Power Mode Protection register, offset: 0x0 */  
    __IO uint8_t PMCTRL;          /**< Power Mode Control register, offset: 0x1 */  
    __IO uint8_t STOPCTRL;         /**< Stop Control Register, offset: 0x2 */  
    __I uint8_t PMSTAT;           /**< Power Mode Status register, offset: 0x3 */  
  
} SMC_Type, *SMC_MemMapPtr;
```

```
/* -----  
-- SMC - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros  
* @{  
*/
```

```

/* SMC - Register accessors */

#define SMC_PMPROT_REG(base)          ((base)->PMPROT)
#define SMC_PMCTRL_REG(base)          ((base)->PMCTRL)
#define SMC_STOPCTRL_REG(base)        ((base)->STOPCTRL)
#define SMC_PMSTAT_REG(base)          ((base)->PMSTAT)

/*!
 * @}
 */

/* /* end of group SMC_Register_Accessor_Macros */

/*
----- --
-- SMC Register Masks
----- */
/*!

 * @addtogroup SMC_Register_Masks SMC Register Masks
 * @{
 */

/* PMPROT Bit Fields */

#define SMC_PMPROT_AVLLS_MASK         0x2u
#define SMC_PMPROT_AVLLS_SHIFT        1
#define SMC_PMPROT_AVLLS_WIDTH        1

#define SMC_PMPROT_AVLLS(x)           (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLLS_SHIFT))&SMC_PMPROT_AVLLS_MASK)

#define SMC_PMPROT_ALLS_MASK          0x8u
#define SMC_PMPROT_ALLS_SHIFT         3

```

```

#define SMC_PMPROT_ALLS_WIDTH          1
#define SMC_PMPROT_ALLS(x)             (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_ALLS_SHIFT))&SMC_PMPROT_ALLS_MASK)
#define SMC_PMPROT_AVLP_MASK           0x20u
#define SMC_PMPROT_AVLP_SHIFT          5
#define SMC_PMPROT_AVLP_WIDTH          1
#define SMC_PMPROT_AVLP(x)             (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLP_SHIFT))&SMC_PMPROT_AVLP_MASK)

/* PMCTRL Bit Fields */

#define SMC_PMCTRL_STOPM_MASK          0x7u
#define SMC_PMCTRL_STOPM_SHIFT         0
#define SMC_PMCTRL_STOPM_WIDTH          3
#define SMC_PMCTRL_STOPM(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPM_SHIFT))&SMC_PMCTRL_STOPM_MASK)
#define SMC_PMCTRL_STOPA_MASK          0x8u
#define SMC_PMCTRL_STOPA_SHIFT         3
#define SMC_PMCTRL_STOPA_WIDTH          1
#define SMC_PMCTRL_STOPA(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPA_SHIFT))&SMC_PMCTRL_STOPA_MASK)
#define SMC_PMCTRL_RUNM_MASK           0x60u
#define SMC_PMCTRL_RUNM_SHIFT          5
#define SMC_PMCTRL_RUNM_WIDTH          2
#define SMC_PMCTRL_RUNM(x)             (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_RUNM_SHIFT))&SMC_PMCTRL_RUNM_MASK)

/* STOPCTRL Bit Fields */

#define SMC_STOPCTRL_VLLSM_MASK        0x7u
#define SMC_STOPCTRL_VLLSM_SHIFT       0
#define SMC_STOPCTRL_VLLSM_WIDTH        3
#define SMC_STOPCTRL_VLLSM(x)          (((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_VLLSM_SHIFT))&SMC_STOPCTRL_VLLSM_MASK)
#define SMC_STOPCTRL_PORPO_MASK        0x20u

```

```

#define SMC_STOPCTRL_PORPO_SHIFT      5
#define SMC_STOPCTRL_PORPO_WIDTH      1
#define SMC_STOPCTRL_PORPO(x)
(((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PORPO_SHIFT))&SMC_STOPCTRL_PORPO_MASK)

#define SMC_STOPCTRL_PSTOPO_MASK      0xC0u
#define SMC_STOPCTRL_PSTOPO_SHIFT     6
#define SMC_STOPCTRL_PSTOPO_WIDTH     2
#define SMC_STOPCTRL_PSTOPO(x)
(((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PSTOPO_SHIFT))&SMC_STOPCTRL_PSTOPO_MASK)

/* PMSTAT Bit Fields */

#define SMC_PMSTAT_PMSTAT_MASK        0x7Fu
#define SMC_PMSTAT_PMSTAT_SHIFT       0
#define SMC_PMSTAT_PMSTAT_WIDTH       7
#define SMC_PMSTAT_PMSTAT(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMSTAT_PMSTAT_SHIFT))&SMC_PMSTAT_PMSTAT_MASK)

/*
 */
/* @}
 */
/* end of group SMC_Register_Masks */

```

```

/* SMC - Peripheral instance base addresses */
/** Peripheral SMC base address */
#define SMC_BASE                      (0x4007E000u)
/** Peripheral SMC base pointer */
#define SMC                         ((SMC_Type *)SMC_BASE)
#define SMC_BASE_PTR                  (SMC)
/** Array initializer of SMC peripheral base addresses */
#define SMC_BASE_ADDRS                { SMC_BASE }
/** Array initializer of SMC peripheral base pointers */

```

```

#define SMC_BASE_PTRS          { SMC }

/*
-----  

-- SMC - Register accessor macros  

----- */
/*!  

* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros  

* @{
* /  

  

/* SMC - Register instance definitions */  

/* SMC */  

#define SMC_PMPROT           SMC_PMPROT_REG(SMC)  

#define SMC_PMCTRL            SMC_PMCTRL_REG(SMC)  

#define SMC_STOPCTRL          SMC_STOPCTRL_REG(SMC)  

#define SMC_PMSTAT             SMC_PMSTAT_REG(SMC)  

  

/*!  

* @}  

* /* end of group SMC_Register_Accessor_Macros */  

  

/*!  

* @}  

* /* end of group SMC_Peripheral_Access_Layer */

```

```

/*
-----  

-- SPI Peripheral Access Layer  

----- */
  
  

/*!  

 * @addtogroup SPI_Peripheral_Access_Layer SPI Peripheral Access Layer
 * @{
 * /  
  

/** SPI - Register Layout Typedef */  

typedef struct {  

    __IO uint8_t C1;           /**< SPI control register 1, offset: 0x0 */  

    __IO uint8_t C2;           /**< SPI control register 2, offset: 0x1 */  

    __IO uint8_t BR;          /**< SPI baud rate register, offset: 0x2 */  

    __IO uint8_t S;            /**< SPI status register, offset: 0x3 */  

    uint8_t RESERVED_0[1];  

    __IO uint8_t D;            /**< SPI data register, offset: 0x5 */  

    uint8_t RESERVED_1[1];  

    __IO uint8_t M;            /**< SPI match register, offset: 0x7 */  

} SPI_Type, *SPI_MemMapPtr;

```

```

/*
-----  

-- SPI - Register accessor macros  

----- */

```

```

/*!  

 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros
 * @{
 * /  


```

```
/* SPI - Register accessors */

#define SPI_C1_REG(base)          ((base)->C1)
#define SPI_C2_REG(base)          ((base)->C2)
#define SPI_BR_REG(base)          ((base)->BR)
#define SPI_S_REG(base)           ((base)->S)
#define SPI_D_REG(base)           ((base)->D)
#define SPI_M_REG(base)           ((base)->M)

/*!
 * @}
 */
/* /* end of group SPI_Register_Accessor_Macros */
```

```
/*
-----  
-- SPI Register Masks  
----- */
```

```
/*!  
 * @addtogroup SPI_Register_Masks SPI Register Masks  
 * @{  
 */
```

```
/* C1 Bit Fields */

#define SPI_C1_LSBFE_MASK        0x1u
#define SPI_C1_LSBFE_SHIFT       0
#define SPI_C1_LSBFE_WIDTH       1
```

```

#define SPI_C1_LSBFE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_LSBFE_SHIFT))&SPI_C1_LSBFE_MASK)

#define SPI_C1_SSOE_MASK          0x2u
#define SPI_C1_SSOE_SHIFT         1
#define SPI_C1_SSOE_WIDTH         1

#define SPI_C1_SSOE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SSOE_SHIFT))&SPI_C1_SSOE_MASK)

#define SPI_C1_CPHA_MASK          0x4u
#define SPI_C1_CPHA_SHIFT         2
#define SPI_C1_CPHA_WIDTH         1

#define SPI_C1_CPHA(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_CPHA_SHIFT))&SPI_C1_CPHA_MASK)

#define SPI_C1_CPOL_MASK          0x8u
#define SPI_C1_CPOL_SHIFT         3
#define SPI_C1_CPOL_WIDTH         1

#define SPI_C1_CPOL(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_CPOL_SHIFT))&SPI_C1_CPOL_MASK)

#define SPI_C1_MSTR_MASK          0x10u
#define SPI_C1_MSTR_SHIFT         4
#define SPI_C1_MSTR_WIDTH         1

#define SPI_C1_MSTR(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_MSTR_SHIFT))&SPI_C1_MSTR_MASK)

#define SPI_C1_SPTIE_MASK          0x20u
#define SPI_C1_SPTIE_SHIFT         5
#define SPI_C1_SPTIE_WIDTH         1

#define SPI_C1_SPTIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPTIE_SHIFT))&SPI_C1_SPTIE_MASK)

#define SPI_C1_SPE_MASK           0x40u
#define SPI_C1_SPE_SHIFT          6
#define SPI_C1_SPE_WIDTH          1

```

```

#define SPI_C1_SPE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPE_SHIFT))&SPI_C1_SPE_MASK)

#define SPI_C1_SPIE_MASK          0x80u
#define SPI_C1_SPIE_SHIFT         7
#define SPI_C1_SPIE_WIDTH         1

#define SPI_C1_SPIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPIE_SHIFT))&SPI_C1_SPIE_MASK)

/* C2 Bit Fields */

#define SPI_C2_SPC0_MASK          0x1u
#define SPI_C2_SPC0_SHIFT         0
#define SPI_C2_SPC0_WIDTH         1

#define SPI_C2_SPC0(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPC0_SHIFT))&SPI_C2_SPC0_MASK)

#define SPI_C2_SPISWAI_MASK        0x2u
#define SPI_C2_SPISWAI_SHIFT       1
#define SPI_C2_SPISWAI_WIDTH       1

#define SPI_C2_SPISWAI(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPISWAI_SHIFT))&SPI_C2_SPISWAI_MASK)

#define SPI_C2_RXDMAE_MASK         0x4u
#define SPI_C2_RXDMAE_SHIFT        2
#define SPI_C2_RXDMAE_WIDTH        1

#define SPI_C2_RXDMAE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_RXDMAE_SHIFT))&SPI_C2_RXDMAE_MASK)

#define SPI_C2_BIDIROE_MASK        0x8u
#define SPI_C2_BIDIROE_SHIFT       3
#define SPI_C2_BIDIROE_WIDTH       1

#define SPI_C2_BIDIROE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_BIDIROE_SHIFT))&SPI_C2_BIDIROE_MASK)

#define SPI_C2_MODFEN_MASK         0x10u
#define SPI_C2_MODFEN_SHIFT        4
#define SPI_C2_MODFEN_WIDTH        1

```

```

#define SPI_C2_MODFEN(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_MODFEN_SHIFT))&SPI_C2_MODFEN_MASK)

#define SPI_C2_TXDMAE_MASK          0x20u
#define SPI_C2_TXDMAE_SHIFT         5
#define SPI_C2_TXDMAE_WIDTH         1

#define SPI_C2_TXDMAE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_TXDMAE_SHIFT))&SPI_C2_TXDMAE_MASK)

#define SPI_C2_SPMIE_MASK           0x80u
#define SPI_C2_SPMIE_SHIFT          7
#define SPI_C2_SPMIE_WIDTH          1

#define SPI_C2_SPMIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPMIE_SHIFT))&SPI_C2_SPMIE_MASK)

/* BR Bit Fields */

#define SPI_BR_SPR_MASK             0xFu
#define SPI_BR_SPR_SHIFT            0
#define SPI_BR_SPR_WIDTH            4

#define SPI_BR_SPR(x)
(((uint8_t)((uint8_t)(x)<<SPI_BR_SPR_SHIFT))&SPI_BR_SPR_MASK)

#define SPI_BR_SPPR_MASK             0x70u
#define SPI_BR_SPPR_SHIFT            4
#define SPI_BR_SPPR_WIDTH            3

#define SPI_BR_SPPR(x)
(((uint8_t)((uint8_t)(x)<<SPI_BR_SPPR_SHIFT))&SPI_BR_SPPR_MASK)

/* S Bit Fields */

#define SPI_S_MODF_MASK              0x10u
#define SPI_S_MODF_SHIFT             4
#define SPI_S_MODF_WIDTH              1

#define SPI_S_MODF(x)
(((uint8_t)((uint8_t)(x)<<SPI_S_MODF_SHIFT))&SPI_S_MODF_MASK)

#define SPI_S_SPTEF_MASK             0x20u
#define SPI_S_SPTEF_SHIFT            5

```

```

#define SPI_S_SPTEF_WIDTH          1
#define SPI_S_SPTEF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPTEF_SHIFT))&SPI_S_SPTEF_MASK)
#define SPI_S_SPMF_MASK            0x40u
#define SPI_S_SPMF_SHIFT           6
#define SPI_S_SPMF_WIDTH           1
#define SPI_S_SPMF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPMF_SHIFT))&SPI_S_SPMF_MASK)
#define SPI_S_SPRF_MASK             0x80u
#define SPI_S_SPRF_SHIFT            7
#define SPI_S_SPRF_WIDTH            1
#define SPI_S_SPRF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPRF_SHIFT))&SPI_S_SPRF_MASK)

/* D Bit Fields */
#define SPI_D_Bits_MASK             0xFFu
#define SPI_D_Bits_SHIFT             0
#define SPI_D_Bits_WIDTH             8
#define SPI_D_Bits(x) (((uint8_t)((uint8_t)(x)<<SPI_D_Bits_SHIFT))&SPI_D_Bits_MASK)

/* M Bit Fields */
#define SPI_M_Bits_MASK             0xFFu
#define SPI_M_Bits_SHIFT             0
#define SPI_M_Bits_WIDTH             8
#define SPI_M_Bits(x) (((uint8_t)((uint8_t)(x)<<SPI_M_Bits_SHIFT))&SPI_M_Bits_MASK)

```

```

/*
 * @}
 */
 /* end of group SPI_Register_Masks */

```

```

/* SPI - Peripheral instance base addresses */

```

```

/** Peripheral SPI0 base address */
#define SPI0_BASE           (0x40076000u)

/** Peripheral SPI0 base pointer */
#define SPI0          ((SPI_Type *)SPI0_BASE)

#define SPI0_BASE_PTR      (SPI0)

/** Peripheral SPI1 base address */
#define SPI1_BASE           (0x40077000u)

/** Peripheral SPI1 base pointer */
#define SPI1          ((SPI_Type *)SPI1_BASE)

#define SPI1_BASE_PTR      (SPI1)

/** Array initializer of SPI peripheral base addresses */
#define SPI_BASE_ADDRS     { SPI0_BASE, SPI1_BASE }

/** Array initializer of SPI peripheral base pointers */
#define SPI_BASE_PTRS       { SPI0, SPI1 }

/*
-----  

-- SPI - Register accessor macros  

----- */
/*!  

 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros  

 * @{
 * /  

  

/* SPI - Register instance definitions */  

/* SPI0 */  

#define SPI0_C1           SPI_C1_REG(SPI0)  

#define SPI0_C2           SPI_C2_REG(SPI0)

```

```

#define SPI0_BR           SPI_BR_REG(SPI0)
#define SPI0_S            SPI_S_REG(SPI0)
#define SPI0_D            SPI_D_REG(SPI0)
#define SPI0_M            SPI_M_REG(SPI0)

/* SPI1 */

#define SPI1_C1           SPI_C1_REG(SPI1)
#define SPI1_C2           SPI_C2_REG(SPI1)
#define SPI1_BR           SPI_BR_REG(SPI1)
#define SPI1_S            SPI_S_REG(SPI1)
#define SPI1_D            SPI_D_REG(SPI1)
#define SPI1_M            SPI_M_REG(SPI1)

/*!
 * @}
 */

/* /* end of group SPI_Register_Accessor_Macros */

/*!
 * @}
 */

/* /* end of group SPI_Peripheral_Access_Layer */

/*
-----*
-- TPM Peripheral Access Layer
----- */
/*!

* @addtogroup TPM_Peripheral_Access_Layer TPM Peripheral Access Layer
* @{

```

```
*/
```

```
/** TPM - Register Layout Typedef */

typedef struct {

    __IO uint32_t SC;           /**< Status and Control, offset: 0x0 */

    __IO uint32_t CNT;          /**< Counter, offset: 0x4 */

    __IO uint32_t MOD;          /**< Modulo, offset: 0x8 */

    struct {                   /* offset: 0xC, array step: 0x8 */

        __IO uint32_t CnSC;      /**< Channel (n) Status and Control, array offset: 0xC, array step: 0x8 */

        __IO uint32_t CnV;       /**< Channel (n) Value, array offset: 0x10, array step: 0x8 */

    } CONTROLS[6];

    uint8_t RESERVED_0[20];

    __IO uint32_t STATUS;       /**< Capture and Compare Status, offset: 0x50 */

    uint8_t RESERVED_1[48];

    __IO uint32_t CONF;         /**< Configuration, offset: 0x84 */

} TPM_Type, *TPM_MemMapPtr;
```

```
/* -----
-- TPM - Register accessor macros
----- */
```

```
/*!
* @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros
* @{
* /
```

```
/* TPM - Register accessors */
```

```
#define TPM_SC_REG(base)          ((base)->SC)
#define TPM_CNT_REG(base)          ((base)->CNT)
#define TPM_MOD_REG(base)          ((base)->MOD)
#define TPM_CnSC_REG(base,index)    ((base)->CONTROLS[index].CnSC)
#define TPM_CnSC_COUNT             6
#define TPM_CnV_REG(base,index)    ((base)->CONTROLS[index].CnV)
#define TPM_CnV_COUNT              6
#define TPM_STATUS_REG(base)        ((base)->STATUS)
#define TPM_CONF_REG(base)          ((base)->CONF)
```

```
/*
 * @}
 */
/* /* end of group TPM_Register_Accessor_Macros */
```

```
/*
-----  
-- TPM Register Masks  
----- */
```

```
/*
 * @addtogroup TPM_Register_Masks TPM Register Masks
 * @{
 */

```

```
/* SC Bit Fields */

#define TPM_SC_PS_MASK           0x7u
#define TPM_SC_PS_SHIFT          0
#define TPM_SC_PS_WIDTH          3
```

```

#define TPM_SC_PS(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_PS_SHIFT))&TPM_SC_PS_MASK)

#define TPM_SC_CMOD_MASK          0x18u
#define TPM_SC_CMOD_SHIFT         3
#define TPM_SC_CMOD_WIDTH         2

#define TPM_SC_CMOD(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_CMOD_SHIFT))&TPM_SC_CMOD_MASK)

#define TPM_SC_CPWMS_MASK         0x20u
#define TPM_SC_CPWMS_SHIFT        5
#define TPM_SC_CPWMS_WIDTH        1

#define TPM_SC_CPWMS(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_CPWMS_SHIFT))&TPM_SC_CPWMS_MASK)

#define TPM_SC_TOIE_MASK          0x40u
#define TPM_SC_TOIE_SHIFT         6
#define TPM_SC_TOIE_WIDTH         1

#define TPM_SC_TOIE(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_TOIE_SHIFT))&TPM_SC_TOIE_MASK)

#define TPM_SC_TOF_MASK           0x80u
#define TPM_SC_TOF_SHIFT          7
#define TPM_SC_TOF_WIDTH          1

#define TPM_SC_TOF(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_TOF_SHIFT))&TPM_SC_TOF_MASK)

#define TPM_SC_DMA_MASK           0x100u
#define TPM_SC_DMA_SHIFT          8
#define TPM_SC_DMA_WIDTH          1

#define TPM_SC_DMA(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_DMA_SHIFT))&TPM_SC_DMA_MASK)

/* CNT Bit Fields */

#define TPM_CNT_COUNT_MASK        0xFFFFu
#define TPM_CNT_COUNT_SHIFT       0
#define TPM_CNT_COUNT_WIDTH      16

```

```

#define TPM_CNT_COUNT(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CNT_COUNT_SHIFT))&TPM_CNT_COUNT_MASK)

/* MOD Bit Fields */

#define TPM_MOD_MOD_MASK          0xFFFFu
#define TPM_MOD_MOD_SHIFT         0
#define TPM_MOD_MOD_WIDTH         16

#define TPM_MOD_MOD(x)
(((uint32_t) (((uint32_t)(x))<<TPM_MOD_MOD_SHIFT))&TPM_MOD_MOD_MASK)

/* CnSC Bit Fields */

#define TPM_CnSC_DMA_MASK         0x1u
#define TPM_CnSC_DMA_SHIFT        0
#define TPM_CnSC_DMA_WIDTH        1

#define TPM_CnSC_DMA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_DMA_SHIFT))&TPM_CnSC_DMA_MASK)

#define TPM_CnSC_ELSA_MASK         0x4u
#define TPM_CnSC_ELSA_SHIFT        2
#define TPM_CnSC_ELSA_WIDTH        1

#define TPM_CnSC_ELSA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_ELSA_SHIFT))&TPM_CnSC_ELSA_MASK)

#define TPM_CnSC_ELSB_MASK         0x8u
#define TPM_CnSC_ELSB_SHIFT        3
#define TPM_CnSC_ELSB_WIDTH        1

#define TPM_CnSC_ELSB(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_ELSB_SHIFT))&TPM_CnSC_ELSB_MASK)

#define TPM_CnSC_MSA_MASK         0x10u
#define TPM_CnSC_MSA_SHIFT        4
#define TPM_CnSC_MSA_WIDTH        1

#define TPM_CnSC_MSA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_MSA_SHIFT))&TPM_CnSC_MSA_MASK)

#define TPM_CnSC_MSB_MASK         0x20u
#define TPM_CnSC_MSB_SHIFT        5

```

```

#define TPM_CnSC_MSB_WIDTH          1
#define TPM_CnSC_MSB(x)             (((uint32_t)((uint32_t)(x)<<TPM_CnSC_MSB_SHIFT))&TPM_CnSC_MSB_MASK)
#define TPM_CnSC_CHIE_MASK          0x40u
#define TPM_CnSC_CHIE_SHIFT         6
#define TPM_CnSC_CHIE_WIDTH         1
#define TPM_CnSC_CHIE(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHIE_SHIFT))&TPM_CnSC_CHIE_MASK)
#define TPM_CnSC_CHF_MASK           0x80u
#define TPM_CnSC_CHF_SHIFT          7
#define TPM_CnSC_CHF_WIDTH          1
#define TPM_CnSC_CHF(x)             (((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHF_SHIFT))&TPM_CnSC_CHF_MASK)

/* CnV Bit Fields */

#define TPM_CnV_VAL_MASK            0xFFFFu
#define TPM_CnV_VAL_SHIFT           0
#define TPM_CnV_VAL_WIDTH          16
#define TPM_CnV_VAL(x)              (((uint32_t)((uint32_t)(x)<<TPM_CnV_VAL_SHIFT))&TPM_CnV_VAL_MASK)

/* STATUS Bit Fields */

#define TPM_STATUS_CH0F_MASK         0x1u
#define TPM_STATUS_CH0F_SHIFT        0
#define TPM_STATUS_CH0F_WIDTH        1
#define TPM_STATUS_CH0F(x)           (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH0F_SHIFT))&TPM_STATUS_CH0F_MASK)
#define TPM_STATUS_CH1F_MASK         0x2u
#define TPM_STATUS_CH1F_SHIFT        1
#define TPM_STATUS_CH1F_WIDTH        1
#define TPM_STATUS_CH1F(x)           (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH1F_SHIFT))&TPM_STATUS_CH1F_MASK)
#define TPM_STATUS_CH2F_MASK         0x4u

```

```

#define TPM_STATUS_CH2F_SHIFT          2
#define TPM_STATUS_CH2F_WIDTH          1
#define TPM_STATUS_CH2F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH2F_SHIFT))&TPM_STATUS_CH2F_MASK)

#define TPM_STATUS_CH3F_MASK           0x8u
#define TPM_STATUS_CH3F_SHIFT          3
#define TPM_STATUS_CH3F_WIDTH          1
#define TPM_STATUS_CH3F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH3F_SHIFT))&TPM_STATUS_CH3F_MASK)

#define TPM_STATUS_CH4F_MASK           0x10u
#define TPM_STATUS_CH4F_SHIFT          4
#define TPM_STATUS_CH4F_WIDTH          1
#define TPM_STATUS_CH4F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH4F_SHIFT))&TPM_STATUS_CH4F_MASK)

#define TPM_STATUS_CH5F_MASK           0x20u
#define TPM_STATUS_CH5F_SHIFT          5
#define TPM_STATUS_CH5F_WIDTH          1
#define TPM_STATUS_CH5F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH5F_SHIFT))&TPM_STATUS_CH5F_MASK)

#define TPM_STATUS_TOF_MASK            0x100u
#define TPM_STATUS_TOF_SHIFT           8
#define TPM_STATUS_TOF_WIDTH           1
#define TPM_STATUS_TOF(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_TOF_SHIFT))&TPM_STATUS_TOF_MASK)

/* CONF Bit Fields */

#define TPM_CONF_DOZEEN_MASK          0x20u
#define TPM_CONF_DOZEEN_SHIFT          5
#define TPM_CONF_DOZEEN_WIDTH          1
#define TPM_CONF_DOZEEN(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_DOZEEN_SHIFT))&TPM_CONF_DOZEEN_MASK)

#define TPM_CONF_DBGMODE_MASK          0xC0u

```

```

#define TPM_CONF_DBGMODE_SHIFT          6
#define TPM_CONF_DBGMODE_WIDTH          2
#define TPM_CONF_DBGMODE(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_DBGMODE_SHIFT))&TPM_CONF_DBGMODE_MASK)

#define TPM_CONF_GTBEEN_MASK           0x200u
#define TPM_CONF_GTBEEN_SHIFT          9
#define TPM_CONF_GTBEEN_WIDTH          1
#define TPM_CONF_GTBEEN(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_GTBEEN_SHIFT))&TPM_CONF_GTBEEN_MASK)

#define TPM_CONF_CSOT_MASK             0x10000u
#define TPM_CONF_CSOT_SHIFT            16
#define TPM_CONF_CSOT_WIDTH            1
#define TPM_CONF_CSOT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOT_SHIFT))&TPM_CONF_CSOT_MASK)

#define TPM_CONF_CSOO_MASK             0x20000u
#define TPM_CONF_CSOO_SHIFT            17
#define TPM_CONF_CSOO_WIDTH            1
#define TPM_CONF_CSOO(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOO_SHIFT))&TPM_CONF_CSOO_MASK)

#define TPM_CONF_CROT_MASK             0x40000u
#define TPM_CONF_CROT_SHIFT            18
#define TPM_CONF_CROT_WIDTH            1
#define TPM_CONF_CROT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CROT_SHIFT))&TPM_CONF_CROT_MASK)

#define TPM_CONF_TRGSEL_MASK           0xF000000u
#define TPM_CONF_TRGSEL_SHIFT          24
#define TPM_CONF_TRGSEL_WIDTH          4
#define TPM_CONF_TRGSEL(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_TRGSEL_SHIFT))&TPM_CONF_TRGSEL_MASK)

```

/*!

```

* @}

/* /* end of group TPM_Register_Masks */

/* TPM - Peripheral instance base addresses */

/** Peripheral TPM0 base address */

#define TPM0_BASE          (0x40038000u)

/** Peripheral TPM0 base pointer */

#define TPM0           ((TPM_Type *)TPM0_BASE)

#define TPM0_BASE_PTR      (TPM0)

/** Peripheral TPM1 base address */

#define TPM1_BASE          (0x40039000u)

/** Peripheral TPM1 base pointer */

#define TPM1           ((TPM_Type *)TPM1_BASE)

#define TPM1_BASE_PTR      (TPM1)

/** Peripheral TPM2 base address */

#define TPM2_BASE          (0x4003A000u)

/** Peripheral TPM2 base pointer */

#define TPM2           ((TPM_Type *)TPM2_BASE)

#define TPM2_BASE_PTR      (TPM2)

/** Array initializer of TPM peripheral base addresses */

#define TPM_BASE_ADDRS     { TPM0_BASE, TPM1_BASE, TPM2_BASE }

/** Array initializer of TPM peripheral base pointers */

#define TPM_BASE_PTRS       { TPM0, TPM1, TPM2 }

/*
----- --
-- TPM - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros  
 * @{  
 */
```

```
/* TPM - Register instance definitions */  
/* TPM0 */  
  
#define TPM0_SC TPM_SC_REG(TPM0)  
#define TPM0_CNT TPM_CNT_REG(TPM0)  
#define TPM0_MOD TPM_MOD_REG(TPM0)  
#define TPM0_COSC TPM_CnSC_REG(TPM0,0)  
#define TPM0_COV TPM_CnV_REG(TPM0,0)  
#define TPM0_C1SC TPM_CnSC_REG(TPM0,1)  
#define TPM0_C1V TPM_CnV_REG(TPM0,1)  
#define TPM0_C2SC TPM_CnSC_REG(TPM0,2)  
#define TPM0_C2V TPM_CnV_REG(TPM0,2)  
#define TPM0_C3SC TPM_CnSC_REG(TPM0,3)  
#define TPM0_C3V TPM_CnV_REG(TPM0,3)  
#define TPM0_C4SC TPM_CnSC_REG(TPM0,4)  
#define TPM0_C4V TPM_CnV_REG(TPM0,4)  
#define TPM0_C5SC TPM_CnSC_REG(TPM0,5)  
#define TPM0_C5V TPM_CnV_REG(TPM0,5)  
#define TPM0_STATUS TPM_STATUS_REG(TPM0)  
#define TPM0_CONF TPM_CONF_REG(TPM0)  
  
/* TPM1 */  
  
#define TPM1_SC TPM_SC_REG(TPM1)  
#define TPM1_CNT TPM_CNT_REG(TPM1)  
#define TPM1_MOD TPM_MOD_REG(TPM1)
```

```

#define TPM1_COSC           TPM_CnSC_REG(TPM1,0)
#define TPM1_COV            TPM_CnV_REG(TPM1,0)
#define TPM1_C1SC           TPM_CnSC_REG(TPM1,1)
#define TPM1_C1V            TPM_CnV_REG(TPM1,1)
#define TPM1_STATUS          TPM_STATUS_REG(TPM1)
#define TPM1_CONF            TPM_CONF_REG(TPM1)

/* TPM2 */

#define TPM2_SC              TPM_SC_REG(TPM2)
#define TPM2_CNT             TPM_CNT_REG(TPM2)
#define TPM2_MOD             TPM_MOD_REG(TPM2)
#define TPM2_COSC           TPM_CnSC_REG(TPM2,0)
#define TPM2_COV            TPM_CnV_REG(TPM2,0)
#define TPM2_C1SC           TPM_CnSC_REG(TPM2,1)
#define TPM2_C1V            TPM_CnV_REG(TPM2,1)
#define TPM2_STATUS          TPM_STATUS_REG(TPM2)
#define TPM2_CONF            TPM_CONF_REG(TPM2)

/* TPM - Register array accessors */

#define TPM0_CnSC(index)     TPM_CnSC_REG(TPM0,index)
#define TPM1_CnSC(index)     TPM_CnSC_REG(TPM1,index)
#define TPM2_CnSC(index)     TPM_CnSC_REG(TPM2,index)
#define TPM0_CnV(index)      TPM_CnV_REG(TPM0,index)
#define TPM1_CnV(index)      TPM_CnV_REG(TPM1,index)
#define TPM2_CnV(index)      TPM_CnV_REG(TPM2,index)

/*
 * @@
 */
 /*/* end of group TPM_Register_Accessor_Macros */

```

```

/*!
* @}
/* /* end of group TPM_Peripheral_Access_Layer */

/*
-----*
-- TSI Peripheral Access Layer
----- */
/*!
* @addtogroup TSI_Peripheral_Access_Layer TSI Peripheral Access Layer
* @{
*/
/* ** TSI - Register Layout Typedef */
typedef struct {
    __IO uint32_t GENCS;           /**< TSI General Control and Status Register, offset: 0x0 */
    __IO uint32_t DATA;            /**< TSI DATA Register, offset: 0x4 */
    __IO uint32_t TSHD;            /**< TSI Threshold Register, offset: 0x8 */
} TSI_Type, *TSI_MemMapPtr;

/*
-----*
-- TSI - Register accessor macros
----- */
/*!
* @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros
* @{

```

```
 */
```

```
/* TSI - Register accessors */

#define TSI_GENCS_REG(base)          ((base)->GENCS)
#define TSI_DATA_REG(base)           ((base)->DATA)
#define TSI_TS HD_REG(base)          ((base)->TS HD)

/*!
 * @}

/* /* end of group TSI_Register_Accessor_Macros */
```

```
/* -----
-- TSI Register Masks
----- */
```

```
/*!
 * @addtogroup TSI_Register_Masks TSI Register Masks
 * @{
 */

/*
```

```
/* GENCS Bit Fields */

#define TSI_GENCS_CURSW_MASK        0x2u
#define TSI_GENCS_CURSW_SHIFT       1
#define TSI_GENCS_CURSW_WIDTH       1

#define TSI_GENCS_CURSW(x)
((uint32_t)((uint32_t)(x)<<TSI_GENCS_CURSW_SHIFT))&TSI_GENCS_CURSW_MASK

#define TSI_GENCS_EOSF_MASK         0x4u
```

```

#define TSI_GENCS_EOSF_SHIFT          2
#define TSI_GENCS_EOSF_WIDTH          1
#define TSI_GENCS_EOSF(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_EOSF_SHIFT))&TSI_GENCS_EOSF_MASK)

#define TSI_GENCS_SCNIP_MASK          0x8u
#define TSI_GENCS_SCNIP_SHIFT         3
#define TSI_GENCS_SCNIP_WIDTH         1
#define TSI_GENCS_SCNIP(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_SCNIP_SHIFT))&TSI_GENCS_SCNIP_MASK)

#define TSI_GENCS_STM_MASK            0x10u
#define TSI_GENCS_STM_SHIFT           4
#define TSI_GENCS_STM_WIDTH           1
#define TSI_GENCS_STM(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_STM_SHIFT))&TSI_GENCS_STM_MASK)

#define TSI_GENCS_STPE_MASK           0x20u
#define TSI_GENCS_STPE_SHIFT          5
#define TSI_GENCS_STPE_WIDTH          1
#define TSI_GENCS_STPE(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_STPE_SHIFT))&TSI_GENCS_STPE_MASK)

#define TSI_GENCS_TSIIEN_MASK          0x40u
#define TSI_GENCS_TSIIEN_SHIFT         6
#define TSI_GENCS_TSIIEN_WIDTH         1
#define TSI_GENCS_TSIIEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIIEN_SHIFT))&TSI_GENCS_TSIIEN_MASK)

#define TSI_GENCS_TSIEN_MASK           0x80u
#define TSI_GENCS_TSIEN_SHIFT          7
#define TSI_GENCS_TSIEN_WIDTH          1
#define TSI_GENCS_TSIEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIEN_SHIFT))&TSI_GENCS_TSIEN_MASK)

#define TSI_GENCS_NSCN_MASK            0x1FO0u
#define TSI_GENCS_NSCN_SHIFT           8

```

```

#define TSI_GENCS_NSCN_WIDTH          5
#define TSI_GENCS_NSCN(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_NSCN_SHIFT))&TSI_GENCS_NSCN_MASK)

#define TSI_GENCS_PS_MASK            0xE000u
#define TSI_GENCS_PS_SHIFT           13
#define TSI_GENCS_PS_WIDTH           3
#define TSI_GENCS_PS(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_PS_SHIFT))&TSI_GENCS_PS_MASK)

#define TSI_GENCS_EXTCHRG_MASK       0x70000u
#define TSI_GENCS_EXTCHRG_SHIFT      16
#define TSI_GENCS_EXTCHRG_WIDTH      3
#define TSI_GENCS_EXTCHRG(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_EXTCHRG_SHIFT))&TSI_GENCS_EXTCHRG_MASK)

#define TSI_GENCS_DVOLT_MASK         0x180000u
#define TSI_GENCS_DVOLT_SHIFT        19
#define TSI_GENCS_DVOLT_WIDTH        2
#define TSI_GENCS_DVOLT(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_DVOLT_SHIFT))&TSI_GENCS_DVOLT_MASK)

#define TSI_GENCS_REFCHRG_MASK       0xE00000u
#define TSI_GENCS_REFCHRG_SHIFT      21
#define TSI_GENCS_REFCHRG_WIDTH      3
#define TSI_GENCS_REFCHRG(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_REFCHRG_SHIFT))&TSI_GENCS_REFCHRG_MASK)

#define TSI_GENCS_MODE_MASK          0xF000000u
#define TSI_GENCS_MODE_SHIFT         24
#define TSI_GENCS_MODE_WIDTH         4
#define TSI_GENCS_MODE(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_MODE_SHIFT))&TSI_GENCS_MODE_MASK)

#define TSI_GENCS_ESOR_MASK          0x10000000u
#define TSI_GENCS_ESOR_SHIFT         28
#define TSI_GENCS_ESOR_WIDTH         1

```

```

#define TSI_GENCS_ESOR(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_ESOR_SHIFT))&TSI_GENCS_ESOR_MASK)

#define TSI_GENCS_OUTRGF_MASK          0x80000000u
#define TSI_GENCS_OUTRGF_SHIFT         31
#define TSI_GENCS_OUTRGF_WIDTH         1

#define TSI_GENCS_OUTRGF(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_OUTRGF_SHIFT))&TSI_GENCS_OUTRGF_MASK)

/* DATA Bit Fields */

#define TSI_DATA_TSICNT_MASK          0xFFFFu
#define TSI_DATA_TSICNT_SHIFT         0
#define TSI_DATA_TSICNT_WIDTH         16

#define TSI_DATA_TSICNT(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICNT_SHIFT))&TSI_DATA_TSICNT_MASK)

#define TSI_DATA_SWTS_MASK            0x400000u
#define TSI_DATA_SWTS_SHIFT           22
#define TSI_DATA_SWTS_WIDTH           1

#define TSI_DATA_SWTS(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_SWTS_SHIFT))&TSI_DATA_SWTS_MASK)

#define TSI_DATA_DMAEN_MASK           0x800000u
#define TSI_DATA_DMAEN_SHIFT          23
#define TSI_DATA_DMAEN_WIDTH          1

#define TSI_DATA_DMAEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_DMAEN_SHIFT))&TSI_DATA_DMAEN_MASK)

#define TSI_DATA_TSICH_MASK           0xF0000000u
#define TSI_DATA_TSICH_SHIFT          28
#define TSI_DATA_TSICH_WIDTH          4

#define TSI_DATA_TSICH(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICH_SHIFT))&TSI_DATA_TSICH_MASK)

/* TSHD Bit Fields */

#define TSI_TSHD_THRESL_MASK          0xFFFFu
#define TSI_TSHD_THRESL_SHIFT          0

```

```

#define TSI_TS HD_THRESL_WIDTH      16
#define TSI_TS HD_THRESL(x)
(((uint32_t) (((uint32_t)(x))<<TSI_TS HD_THRESL_SHIFT))&TSI_TS HD_THRESL_MASK)

#define TSI_TS HD_THRESH_MASK        0xFFFF0000u
#define TSI_TS HD_THRESH_SHIFT      16
#define TSI_TS HD_THRESH_WIDTH      16
#define TSI_TS HD_THRESH(x)
(((uint32_t) (((uint32_t)(x))<<TSI_TS HD_THRESH_SHIFT))&TSI_TS HD_THRESH_MASK)

/*
 * @}
 */

/* /* end of group TSI_Register_Masks */

/* TSI - Peripheral instance base addresses */
/** Peripheral TSI0 base address */
#define TSI0_BASE                  (0x40045000u)
/** Peripheral TSI0 base pointer */
#define TSI0 ((TSI_Type *)TSI0_BASE)
#define TSI0_BASE_PTR              (TSI0)
/** Array initializer of TSI peripheral base addresses */
#define TSI_BASE_ADDRS             { TSI0_BASE }
/** Array initializer of TSI peripheral base pointers */
#define TSI_BASE_PTRS              { TSI0 }

/*
-----*
-- TSI - Register accessor macros
-----*/

```

```
/*!  
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros  
 * @{  
 */
```

```
/* TSI - Register instance definitions */  
/* TSI0 */  
  
#define TSI0_GENCS          TSI_GENCS_REG(TSI0)  
#define TSI0_DATA           TSI_DATA_REG(TSI0)  
#define TSI0_TS HD          TSI_TS HD_REG(TSI0)
```

```
/*!  
 * @}  
 */ /* end of group TSI_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group TSI_Peripheral_Access_Layer */
```

```
/* -----  
-- UART Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup UART_Peripheral_Access_Layer UART Peripheral Access Layer  
 * @{
```

```
 */
```

```
/** UART - Register Layout Typedef */

typedef struct {

    __IO uint8_t BDH;           /**< UART Baud Rate Register: High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register: Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __I  uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;              /**< UART Data Register, offset: 0x7 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0x8 */

} UART_Type, *UART_MemMapPtr;
```

```
/*
-----  
-- UART - Register accessor macros  
----- */
```

```
/*!
```

```
* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros  
* @{  
*/
```

```
/* UART - Register accessors */  
  
#define UART_BDH_REG(base)      ((base)->BDH)  
  
#define UART_BDL_REG(base)      ((base)->BDL)  
  
#define UART_C1_REG(base)       ((base)->C1)
```

```
#define UART_C2_REG(base)          ((base)->C2)
#define UART_S1_REG(base)           ((base)->S1)
#define UART_S2_REG(base)           ((base)->S2)
#define UART_C3_REG(base)           ((base)->C3)
#define UART_D_REG(base)            ((base)->D)
#define UART_C4_REG(base)           ((base)->C4)
```

```
/*!
 * @}
 */
/* /* end of group UART_Register_Accessor_Macros */
```

```
/*
-----  
-- UART Register Masks  
----- */
```

```
/*!
 * @addtogroup UART_Register_Masks UART Register Masks
 * @{
 */

```

```
/* BDH Bit Fields */

#define UART_BDH_SBR_MASK          0x1Fu
#define UART_BDH_SBR_SHIFT         0
#define UART_BDH_SBR_WIDTH          5

#define UART_BDH_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART_BDH_SBR_SHIFT))&UART_BDH_SBR_MASK)

#define UART_BDH_SBNS_MASK         0x20u
#define UART_BDH_SBNS_SHIFT          5
```

```

#define UART_BDH_SBNS_WIDTH          1
#define UART_BDH_SBNS(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_SBNS_SHIFT))&UART_BDH_SBNS_MASK)
#define UART_BDH_RXEDGIE_MASK        0x40u
#define UART_BDH_RXEDGIE_SHIFT       6
#define UART_BDH_RXEDGIE_WIDTH       1
#define UART_BDH_RXEDGIE(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_RXEDGIE_SHIFT))&UART_BDH_RXEDGIE_MASK)
#define UART_BDH_LBKDIE_MASK         0x80u
#define UART_BDH_LBKDIE_SHIFT        7
#define UART_BDH_LBKDIE_WIDTH        1
#define UART_BDH_LBKDIE(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_LBKDIE_SHIFT))&UART_BDH_LBKDIE_MASK)
/* BDL Bit Fields */
#define UART_BDL_SBR_MASK           0xFFu
#define UART_BDL_SBR_SHIFT          0
#define UART_BDL_SBR_WIDTH          8
#define UART_BDL_SBR(x) (((uint8_t)((uint8_t)(x)<<UART_BDL_SBR_SHIFT))&UART_BDL_SBR_MASK)
/* C1 Bit Fields */
#define UART_C1_PT_MASK             0x1u
#define UART_C1_PT_SHIFT             0
#define UART_C1_PT_WIDTH             1
#define UART_C1_PT(x) (((uint8_t)((uint8_t)(x)<<UART_C1_PT_SHIFT))&UART_C1_PT_MASK)
#define UART_C1_PE_MASK              0x2u
#define UART_C1_PE_SHIFT             1
#define UART_C1_PE_WIDTH             1
#define UART_C1_PE(x) (((uint8_t)((uint8_t)(x)<<UART_C1_PE_SHIFT))&UART_C1_PE_MASK)
#define UART_C1_ILT_MASK             0x4u

```

```

#define UART_C1_ILT_SHIFT          2
#define UART_C1_ILT_WIDTH          1
#define UART_C1_ILT(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_ILT_SHIFT))&UART_C1_ILT_MASK)

#define UART_C1_WAKE_MASK          0x8u
#define UART_C1_WAKE_SHIFT          3
#define UART_C1_WAKE_WIDTH          1
#define UART_C1_WAKE(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_WAKE_SHIFT))&UART_C1_WAKE_MASK)

#define UART_C1_M_MASK              0x10u
#define UART_C1_M_SHIFT             4
#define UART_C1_M_WIDTH              1
#define UART_C1_M(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_M_SHIFT))&UART_C1_M_MASK)

#define UART_C1_RSRC_MASK           0x20u
#define UART_C1_RSRC_SHIFT           5
#define UART_C1_RSRC_WIDTH           1
#define UART_C1_RSRC(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_RSRC_SHIFT))&UART_C1_RSRC_MASK)

#define UART_C1_UARTSWAI_MASK        0x40u
#define UART_C1_UARTSWAI_SHIFT        6
#define UART_C1_UARTSWAI_WIDTH        1
#define UART_C1_UARTSWAI(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_UARTSWAI_SHIFT))&UART_C1_UARTSWAI_MASK)

#define UART_C1_LOOPS_MASK           0x80u
#define UART_C1_LOOPS_SHIFT           7
#define UART_C1_LOOPS_WIDTH           1
#define UART_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_LOOPS_SHIFT))&UART_C1_LOOPS_MASK)

/* C2 Bit Fields */

#define UART_C2_SBK_MASK             0x1u

```

```

#define UART_C2_SBK_SHIFT          0
#define UART_C2_SBK_WIDTH          1
#define UART_C2_SBK(x) (((uint8_t)((uint8_t)(x)<<UART_C2_SBK_SHIFT))&UART_C2_SBK_MASK)

#define UART_C2_RWU_MASK           0x2u
#define UART_C2_RWU_SHIFT          1
#define UART_C2_RWU_WIDTH          1
#define UART_C2_RWU(x) (((uint8_t)((uint8_t)(x)<<UART_C2_RWU_SHIFT))&UART_C2_RWU_MASK)

#define UART_C2_RE_MASK            0x4u
#define UART_C2_RE_SHIFT           2
#define UART_C2_RE_WIDTH           1
#define UART_C2_RE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_RE_SHIFT))&UART_C2_RE_MASK)

#define UART_C2_TE_MASK            0x8u
#define UART_C2_TE_SHIFT           3
#define UART_C2_TE_WIDTH           1
#define UART_C2_TE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TE_SHIFT))&UART_C2_TE_MASK)

#define UART_C2_ILIE_MASK          0x10u
#define UART_C2_ILIE_SHIFT          4
#define UART_C2_ILIE_WIDTH          1
#define UART_C2_ILIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_ILIE_SHIFT))&UART_C2_ILIE_MASK)

#define UART_C2_RIE_MASK           0x20u
#define UART_C2_RIE_SHIFT          5
#define UART_C2_RIE_WIDTH           1
#define UART_C2_RIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_RIE_SHIFT))&UART_C2_RIE_MASK)

#define UART_C2_TCIE_MASK          0x40u
#define UART_C2_TCIE_SHIFT          6

```

```

#define UART_C2_TCIE_WIDTH           1
#define UART_C2_TCIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TCIE_SHIFT))&UART_C2_TCIE_MASK)
#define UART_C2_TIE_MASK             0x80u
#define UART_C2_TIE_SHIFT            7
#define UART_C2_TIE_WIDTH            1
#define UART_C2_TIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TIE_SHIFT))&UART_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART_S1_PF_MASK              0x1u
#define UART_S1_PF_SHIFT              0
#define UART_S1_PF_WIDTH              1
#define UART_S1_PF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_PF_SHIFT))&UART_S1_PF_MASK)

#define UART_S1_FE_MASK              0x2u
#define UART_S1_FE_SHIFT              1
#define UART_S1_FE_WIDTH              1
#define UART_S1_FE(x) (((uint8_t)((uint8_t)(x)<<UART_S1_FE_SHIFT))&UART_S1_FE_MASK)

#define UART_S1_NF_MASK              0x4u
#define UART_S1_NF_SHIFT              2
#define UART_S1_NF_WIDTH              1
#define UART_S1_NF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_NF_SHIFT))&UART_S1_NF_MASK)

#define UART_S1_OR_MASK              0x8u
#define UART_S1_OR_SHIFT              3
#define UART_S1_OR_WIDTH              1
#define UART_S1_OR(x) (((uint8_t)((uint8_t)(x)<<UART_S1_OR_SHIFT))&UART_S1_OR_MASK)

#define UART_S1_IDLE_MASK             0x10u
#define UART_S1_IDLE_SHIFT             4

```

```

#define UART_S1_IDLE_WIDTH          1
#define UART_S1_IDLE(x)             (((uint8_t)((uint8_t)(x))<<UART_S1_IDLE_SHIFT))&UART_S1_IDLE_MASK
#define UART_S1_RDRF_MASK           0x20u
#define UART_S1_RDRF_SHIFT          5
#define UART_S1_RDRF_WIDTH          1
#define UART_S1_RDRF(x)             (((uint8_t)((uint8_t)(x))<<UART_S1_RDRF_SHIFT))&UART_S1_RDRF_MASK
#define UART_S1_TC_MASK              0x40u
#define UART_S1_TC_SHIFT             6
#define UART_S1_TC_WIDTH             1
#define UART_S1_TC(x)               (((uint8_t)((uint8_t)(x))<<UART_S1_TC_SHIFT))&UART_S1_TC_MASK
#define UART_S1_TDRE_MASK            0x80u
#define UART_S1_TDRE_SHIFT           7
#define UART_S1_TDRE_WIDTH           1
#define UART_S1_TDRE(x)              (((uint8_t)((uint8_t)(x))<<UART_S1_TDRE_SHIFT))&UART_S1_TDRE_MASK
/* S2 Bit Fields */
#define UART_S2_RAF_MASK              0x1u
#define UART_S2_RAF_SHIFT             0
#define UART_S2_RAF_WIDTH             1
#define UART_S2_RAF(x)                (((uint8_t)((uint8_t)(x))<<UART_S2_RAF_SHIFT))&UART_S2_RAF_MASK
#define UART_S2_LBKDE_MASK             0x2u
#define UART_S2_LBKDE_SHIFT            1
#define UART_S2_LBKDE_WIDTH           1
#define UART_S2_LBKDE(x)               (((uint8_t)((uint8_t)(x))<<UART_S2_LBKDE_SHIFT))&UART_S2_LBKDE_MASK
#define UART_S2_BRK13_MASK              0x4u
#define UART_S2_BRK13_SHIFT             2

```

```

#define UART_S2_BRK13_WIDTH          1
#define UART_S2_BRK13(x) (((uint8_t)((uint8_t)(x)<<UART_S2_BRK13_SHIFT))&UART_S2_BRK13_MASK)
#define UART_S2_RWUID_MASK           0x8u
#define UART_S2_RWUID_SHIFT          3
#define UART_S2_RWUID_WIDTH          1
#define UART_S2_RWUID(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RWUID_SHIFT))&UART_S2_RWUID_MASK)
#define UART_S2_RXINV_MASK           0x10u
#define UART_S2_RXINV_SHIFT          4
#define UART_S2_RXINV_WIDTH          1
#define UART_S2_RXINV(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RXINV_SHIFT))&UART_S2_RXINV_MASK)
#define UART_S2_RXEDGIF_MASK          0x40u
#define UART_S2_RXEDGIF_SHIFT         6
#define UART_S2_RXEDGIF_WIDTH         1
#define UART_S2_RXEDGIF(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RXEDGIF_SHIFT))&UART_S2_RXEDGIF_MASK)
#define UART_S2_LBKDIF_MASK           0x80u
#define UART_S2_LBKDIF_SHIFT          7
#define UART_S2_LBKDIF_WIDTH          1
#define UART_S2_LBKDIF(x) (((uint8_t)((uint8_t)(x)<<UART_S2_LBKDIF_SHIFT))&UART_S2_LBKDIF_MASK)
/* C3 Bit Fields */
#define UART_C3_PEIE_MASK            0x1u
#define UART_C3_PEIE_SHIFT             0
#define UART_C3_PEIE_WIDTH             1
#define UART_C3_PEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_PEIE_SHIFT))&UART_C3_PEIE_MASK)
#define UART_C3_FEIE_MASK             0x2u
#define UART_C3_FEIE_SHIFT              1

```

```

#define UART_C3_FEIE_WIDTH          1
#define UART_C3_FEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_FEIE_SHIFT))&UART_C3_FEIE_MASK)
#define UART_C3_NEIE_MASK           0x4u
#define UART_C3_NEIE_SHIFT          2
#define UART_C3_NEIE_WIDTH          1
#define UART_C3_NEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_NEIE_SHIFT))&UART_C3_NEIE_MASK)
#define UART_C3_ORIE_MASK           0x8u
#define UART_C3_ORIE_SHIFT          3
#define UART_C3_ORIE_WIDTH          1
#define UART_C3_ORIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_ORIE_SHIFT))&UART_C3_ORIE_MASK)
#define UART_C3_TXINV_MASK           0x10u
#define UART_C3_TXINV_SHIFT          4
#define UART_C3_TXINV_WIDTH          1
#define UART_C3_TXINV(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXINV_SHIFT))&UART_C3_TXINV_MASK)
#define UART_C3_TXDIR_MASK           0x20u
#define UART_C3_TXDIR_SHIFT          5
#define UART_C3_TXDIR_WIDTH          1
#define UART_C3_TXDIR(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXDIR_SHIFT))&UART_C3_TXDIR_MASK)
#define UART_C3_T8_MASK              0x40u
#define UART_C3_T8_SHIFT             6
#define UART_C3_T8_WIDTH             1
#define UART_C3_T8(x) (((uint8_t)((uint8_t)(x)<<UART_C3_T8_SHIFT))&UART_C3_T8_MASK)
#define UART_C3_R8_MASK              0x80u
#define UART_C3_R8_SHIFT             7
#define UART_C3_R8_WIDTH             1

```

```

#define UART_C3_R8(x)
(((uint8_t)((uint8_t)(x)<<UART_C3_R8_SHIFT))&UART_C3_R8_MASK)

/* D Bit Fields */

#define UART_DROTO_MASK          0x1u
#define UART_DROTO_SHIFT         0
#define UART_DROTO_WIDTH         1

#define UART_DROTO(x)
(((uint8_t)((uint8_t)(x)<<UART_DROTO_SHIFT))&UART_DROTO_MASK)

#define UART_D_R1T1_MASK          0x2u
#define UART_D_R1T1_SHIFT         1
#define UART_D_R1T1_WIDTH         1

#define UART_D_R1T1(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R1T1_SHIFT))&UART_D_R1T1_MASK)

#define UART_D_R2T2_MASK          0x4u
#define UART_D_R2T2_SHIFT         2
#define UART_D_R2T2_WIDTH         1

#define UART_D_R2T2(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R2T2_SHIFT))&UART_D_R2T2_MASK)

#define UART_D_R3T3_MASK          0x8u
#define UART_D_R3T3_SHIFT         3
#define UART_D_R3T3_WIDTH         1

#define UART_D_R3T3(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R3T3_SHIFT))&UART_D_R3T3_MASK)

#define UART_D_R4T4_MASK          0x10u
#define UART_D_R4T4_SHIFT         4
#define UART_D_R4T4_WIDTH         1

#define UART_D_R4T4(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R4T4_SHIFT))&UART_D_R4T4_MASK)

#define UART_D_R5T5_MASK          0x20u
#define UART_D_R5T5_SHIFT         5
#define UART_D_R5T5_WIDTH         1

```

```

#define UART_D_R5T5(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R5T5_SHIFT))&UART_D_R5T5_MASK)

#define UART_D_R6T6_MASK          0x40u
#define UART_D_R6T6_SHIFT         6
#define UART_D_R6T6_WIDTH         1

#define UART_D_R6T6(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R6T6_SHIFT))&UART_D_R6T6_MASK)

#define UART_D_R7T7_MASK          0x80u
#define UART_D_R7T7_SHIFT         7
#define UART_D_R7T7_WIDTH         1

#define UART_D_R7T7(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R7T7_SHIFT))&UART_D_R7T7_MASK)

/* C4 Bit Fields */

#define UART_C4_RDMAS_MASK        0x20u
#define UART_C4_RDMAS_SHIFT       5
#define UART_C4_RDMAS_WIDTH       1

#define UART_C4_RDMAS(x)
(((uint8_t)((uint8_t)(x)<<UART_C4_RDMAS_SHIFT))&UART_C4_RDMAS_MASK)

#define UART_C4_TDMAS_MASK        0x80u
#define UART_C4_TDMAS_SHIFT       7
#define UART_C4_TDMAS_WIDTH       1

#define UART_C4_TDMAS(x)
(((uint8_t)((uint8_t)(x)<<UART_C4_TDMAS_SHIFT))&UART_C4_TDMAS_MASK)

/*
 * @}
 */

/* /* end of group UART_Register_Masks */

/* UART - Peripheral instance base addresses */

```

```

/** Peripheral UART1 base address */
#define UART1_BASE           (0x4006B000u)

/** Peripheral UART1 base pointer */
#define UART1             ((UART_Type *)UART1_BASE)
#define UART1_BASE_PTR      (UART1)

/** Peripheral UART2 base address */
#define UART2_BASE           (0x4006C000u)

/** Peripheral UART2 base pointer */
#define UART2             ((UART_Type *)UART2_BASE)
#define UART2_BASE_PTR      (UART2)

/** Array initializer of UART peripheral base addresses */
#define UART_BASE_ADDRS     { UART1_BASE, UART2_BASE }

/** Array initializer of UART peripheral base pointers */
#define UART_BASE_PTRS       { UART1, UART2 }

```

```

/*
-----  

-- UART - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros  

* @{
* /  


```

```

/* UART - Register instance definitions */  

/* UART1 */  

#define UART1_BDH           UART_BDH_REG(UART1)  

#define UART1_BDL           UART_BDL_REG(UART1)

```

```
#define UART1_C1          UART_C1_REG(UART1)
#define UART1_C2          UART_C2_REG(UART1)
#define UART1_S1          UART_S1_REG(UART1)
#define UART1_S2          UART_S2_REG(UART1)
#define UART1_C3          UART_C3_REG(UART1)
#define UART1_D           UART_D_REG(UART1)
#define UART1_C4          UART_C4_REG(UART1)

/* UART2 */

#define UART2_BDH         UART_BDH_REG(UART2)
#define UART2_BDL         UART_BDL_REG(UART2)
#define UART2_C1          UART_C1_REG(UART2)
#define UART2_C2          UART_C2_REG(UART2)
#define UART2_S1          UART_S1_REG(UART2)
#define UART2_S2          UART_S2_REG(UART2)
#define UART2_C3          UART_C3_REG(UART2)
#define UART2_D           UART_D_REG(UART2)
#define UART2_C4          UART_C4_REG(UART2)

/*!
 * @}
 */
/* /* end of group UART_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* /* end of group UART_Peripheral_Access_Layer */
```

```
/* -----
```

```

-- UART0 Peripheral Access Layer
----- */
/*!
* @addtogroup UART0_Peripheral_Access_Layer UART0 Peripheral Access Layer
* @{
*/
/** UART0 - Register Layout Typedef */
typedef struct {
    __IO uint8_t BDH;           /**< UART Baud Rate Register High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __IO uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;             /**< UART Data Register, offset: 0x7 */
    __IO uint8_t MA1;           /**< UART Match Address Registers 1, offset: 0x8 */
    __IO uint8_t MA2;           /**< UART Match Address Registers 2, offset: 0x9 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0xA */
    __IO uint8_t C5;            /**< UART Control Register 5, offset: 0xB */
} UART0_Type, *UART0_MemMapPtr;

/*
----- --
-- UART0 - Register accessor macros
----- */
/*!

```

```
* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros
* @{
* /
```

```
/* UART0 - Register accessors */

#define UART0_BDH_REG(base)          ((base)->BDH)
#define UART0_BDL_REG(base)          ((base)->BDL)
#define UART0_C1_REG(base)           ((base)->C1)
#define UART0_C2_REG(base)           ((base)->C2)
#define UART0_S1_REG(base)           ((base)->S1)
#define UART0_S2_REG(base)           ((base)->S2)
#define UART0_C3_REG(base)           ((base)->C3)
#define UART0_D_REG(base)            ((base)->D)
#define UART0_MA1_REG(base)          ((base)->MA1)
#define UART0_MA2_REG(base)          ((base)->MA2)
#define UART0_C4_REG(base)           ((base)->C4)
#define UART0_C5_REG(base)           ((base)->C5)
```

```
/*
* @@
* /* end of group UART0_Register_Accessor_Macros */
```

```
/* -----
-- UART0 Register Masks
----- */
```

```
/*!
```

```

* @addtogroup UART0_Register_Masks UART0 Register Masks

* @{
*

/* BDH Bit Fields */

#define UART0_BDH_SBR_MASK          0x1Fu
#define UART0_BDH_SBR_SHIFT         0
#define UART0_BDH_SBR_WIDTH         5
#define UART0_BDH_SBR(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_SBR_SHIFT))&UART0_BDH_SBR_MASK)

#define UART0_BDH_SBNS_MASK         0x20u
#define UART0_BDH_SBNS_SHIFT        5
#define UART0_BDH_SBNS_WIDTH        1
#define UART0_BDH_SBNS(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_SBNS_SHIFT))&UART0_BDH_SBNS_MASK)

#define UART0_BDH_RXEDGIE_MASK      0x40u
#define UART0_BDH_RXEDGIE_SHIFT     6
#define UART0_BDH_RXEDGIE_WIDTH     1
#define UART0_BDH_RXEDGIE(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_RXEDGIE_SHIFT))&UART0_BDH_RXEDGIE_MASK)

#define UART0_BDH_LBKDIE_MASK       0x80u
#define UART0_BDH_LBKDIE_SHIFT      7
#define UART0_BDH_LBKDIE_WIDTH      1
#define UART0_BDH_LBKDIE(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_LBKDIE_SHIFT))&UART0_BDH_LBKDIE_MASK)

/* BDL Bit Fields */

#define UART0_BDL_SBR_MASK          0xFFu
#define UART0_BDL_SBR_SHIFT         0
#define UART0_BDL_SBR_WIDTH         8

```

```

#define UART0_BDL_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART0_BDL_SBR_SHIFT))&UART0_BDL_SBR_MASK)

/* C1 Bit Fields */

#define UART0_C1_PT_MASK          0x1u
#define UART0_C1_PT_SHIFT         0
#define UART0_C1_PT_WIDTH         1

#define UART0_C1_PT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PT_SHIFT))&UART0_C1_PT_MASK)

#define UART0_C1_PE_MASK          0x2u
#define UART0_C1_PE_SHIFT         1
#define UART0_C1_PE_WIDTH         1

#define UART0_C1_PE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PE_SHIFT))&UART0_C1_PE_MASK)

#define UART0_C1_ILT_MASK          0x4u
#define UART0_C1_ILT_SHIFT         2
#define UART0_C1_ILT_WIDTH         1

#define UART0_C1_ILT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_ILT_SHIFT))&UART0_C1_ILT_MASK)

#define UART0_C1_WAKE_MASK          0x8u
#define UART0_C1_WAKE_SHIFT         3
#define UART0_C1_WAKE_WIDTH         1

#define UART0_C1_WAKE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_WAKE_SHIFT))&UART0_C1_WAKE_MASK)

#define UART0_C1_M_MASK           0x10u
#define UART0_C1_M_SHIFT            4
#define UART0_C1_M_WIDTH             1

#define UART0_C1_M(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_M_SHIFT))&UART0_C1_M_MASK)

#define UART0_C1_RSRC_MASK          0x20u
#define UART0_C1_RSRC_SHIFT         5
#define UART0_C1_RSRC_WIDTH         1

```

```

#define UART0_C1_RSRC(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_RSRC_SHIFT))&UART0_C1_RSRC_MASK)

#define UART0_C1_DOZEEN_MASK          0x40u
#define UART0_C1_DOZEEN_SHIFT        6
#define UART0_C1_DOZEEN_WIDTH        1

#define UART0_C1_DOZEEN(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_DOZEEN_SHIFT))&UART0_C1_DOZEEN_MASK)

#define UART0_C1_LOOPS_MASK          0x80u
#define UART0_C1_LOOPS_SHIFT        7
#define UART0_C1_LOOPS_WIDTH        1

#define UART0_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_LOOPS_SHIFT))&UART0_C1_LOOPS_MASK)

/* C2 Bit Fields */

#define UART0_C2_SBK_MASK           0x1u
#define UART0_C2_SBK_SHIFT          0
#define UART0_C2_SBK_WIDTH          1

#define UART0_C2_SBK(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_SBK_SHIFT))&UART0_C2_SBK_MASK)

#define UART0_C2_RWU_MASK           0x2u
#define UART0_C2_RWU_SHIFT          1
#define UART0_C2_RWU_WIDTH          1

#define UART0_C2_RWU(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RWU_SHIFT))&UART0_C2_RWU_MASK)

#define UART0_C2_RE_MASK            0x4u
#define UART0_C2_RE_SHIFT           2
#define UART0_C2_RE_WIDTH           1

#define UART0_C2_RE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RE_SHIFT))&UART0_C2_RE_MASK)

#define UART0_C2_TE_MASK            0x8u
#define UART0_C2_TE_SHIFT           3
#define UART0_C2_TE_WIDTH           1

```

```

#define UART0_C2_TE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TE_SHIFT))&UART0_C2_TE_MASK)

#define UART0_C2_ILIE_MASK          0x10u
#define UART0_C2_ILIE_SHIFT         4
#define UART0_C2_ILIE_WIDTH         1

#define UART0_C2_ILIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_ILIE_SHIFT))&UART0_C2_ILIE_MASK)

#define UART0_C2_RIE_MASK           0x20u
#define UART0_C2_RIE_SHIFT          5
#define UART0_C2_RIE_WIDTH          1

#define UART0_C2_RIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_RIE_SHIFT))&UART0_C2_RIE_MASK)

#define UART0_C2_TCIE_MASK          0x40u
#define UART0_C2_TCIE_SHIFT          6
#define UART0_C2_TCIE_WIDTH          1

#define UART0_C2_TCIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TCIE_SHIFT))&UART0_C2_TCIE_MASK)

#define UART0_C2_TIE_MASK            0x80u
#define UART0_C2_TIE_SHIFT           7
#define UART0_C2_TIE_WIDTH           1

#define UART0_C2_TIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TIE_SHIFT))&UART0_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART0_S1_PF_MASK             0x1u
#define UART0_S1_PF_SHIFT              0
#define UART0_S1_PF_WIDTH              1

#define UART0_S1_PF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_PF_SHIFT))&UART0_S1_PF_MASK)

#define UART0_S1_FE_MASK              0x2u
#define UART0_S1_FE_SHIFT                1
#define UART0_S1_FE_WIDTH                1

```

```

#define UART0_S1_FE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_FE_SHIFT))&UART0_S1_FE_MASK)

#define UART0_S1_NF_MASK          0x4u
#define UART0_S1_NF_SHIFT         2
#define UART0_S1_NF_WIDTH         1

#define UART0_S1_NF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_NF_SHIFT))&UART0_S1_NF_MASK)

#define UART0_S1_OR_MASK          0x8u
#define UART0_S1_OR_SHIFT         3
#define UART0_S1_OR_WIDTH         1

#define UART0_S1_OR(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_OR_SHIFT))&UART0_S1_OR_MASK)

#define UART0_S1_IDLE_MASK         0x10u
#define UART0_S1_IDLE_SHIFT        4
#define UART0_S1_IDLE_WIDTH        1

#define UART0_S1_IDLE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_IDLE_SHIFT))&UART0_S1_IDLE_MASK)

#define UART0_S1_RDRF_MASK         0x20u
#define UART0_S1_RDRF_SHIFT        5
#define UART0_S1_RDRF_WIDTH        1

#define UART0_S1_RDRF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_RDRF_SHIFT))&UART0_S1_RDRF_MASK)

#define UART0_S1_TC_MASK           0x40u
#define UART0_S1_TC_SHIFT          6
#define UART0_S1_TC_WIDTH          1

#define UART0_S1_TC(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TC_SHIFT))&UART0_S1_TC_MASK)

#define UART0_S1_TDRE_MASK          0x80u
#define UART0_S1_TDRE_SHIFT         7
#define UART0_S1_TDRE_WIDTH         1

```

```

#define UART0_S1_TDRE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TDRE_SHIFT))&UART0_S1_TDRE_MASK)

/* S2 Bit Fields */

#define UART0_S2_RAF_MASK          0x1u
#define UART0_S2_RAF_SHIFT         0
#define UART0_S2_RAF_WIDTH         1

#define UART0_S2_RAF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RAF_SHIFT))&UART0_S2_RAF_MASK)

#define UART0_S2_LBKDE_MASK        0x2u
#define UART0_S2_LBKDE_SHIFT       1
#define UART0_S2_LBKDE_WIDTH       1

#define UART0_S2_LBKDE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_LBKDE_SHIFT))&UART0_S2_LBKDE_MASK)

#define UART0_S2_BRK13_MASK         0x4u
#define UART0_S2_BRK13_SHIFT        2
#define UART0_S2_BRK13_WIDTH        1

#define UART0_S2_BRK13(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_BRK13_SHIFT))&UART0_S2_BRK13_MASK)

#define UART0_S2_RWUID_MASK         0x8u
#define UART0_S2_RWUID_SHIFT        3
#define UART0_S2_RWUID_WIDTH        1

#define UART0_S2_RWUID(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RWUID_SHIFT))&UART0_S2_RWUID_MASK)

#define UART0_S2_RXINV_MASK         0x10u
#define UART0_S2_RXINV_SHIFT        4
#define UART0_S2_RXINV_WIDTH        1

#define UART0_S2_RXINV(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RXINV_SHIFT))&UART0_S2_RXINV_MASK)

#define UART0_S2_MSBF_MASK          0x20u
#define UART0_S2_MSBF_SHIFT         5
#define UART0_S2_MSBF_WIDTH         1

```

```

#define UART0_S2_MSBF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_MSBF_SHIFT))&UART0_S2_MSBF_MASK

#define UART0_S2_RXEDGIF_MASK          0x40u
#define UART0_S2_RXEDGIF_SHIFT        6
#define UART0_S2_RXEDGIF_WIDTH        1

#define UART0_S2_RXEDGIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_RXEDGIF_SHIFT))&UART0_S2_RXEDGIF_MASK

#define UART0_S2_LBKDIF_MASK          0x80u
#define UART0_S2_LBKDIF_SHIFT        7
#define UART0_S2_LBKDIF_WIDTH        1

#define UART0_S2_LBKDIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_LBKDIF_SHIFT))&UART0_S2_LBKDIF_MASK

/* C3 Bit Fields */

#define UART0_C3_PEIE_MASK           0x1u
#define UART0_C3_PEIE_SHIFT          0
#define UART0_C3_PEIE_WIDTH          1

#define UART0_C3_PEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_PEIE_SHIFT))&UART0_C3_PEIE_MASK

#define UART0_C3_FEIE_MASK           0x2u
#define UART0_C3_FEIE_SHIFT          1
#define UART0_C3_FEIE_WIDTH          1

#define UART0_C3_FEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_FEIE_SHIFT))&UART0_C3_FEIE_MASK

#define UART0_C3_NEIE_MASK           0x4u
#define UART0_C3_NEIE_SHIFT          2
#define UART0_C3_NEIE_WIDTH          1

#define UART0_C3_NEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_NEIE_SHIFT))&UART0_C3_NEIE_MASK

#define UART0_C3_ORIE_MASK           0x8u
#define UART0_C3_ORIE_SHIFT          3
#define UART0_C3_ORIE_WIDTH          1

```

```

#define UART0_C3_ORIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_ORIE_SHIFT))&UART0_C3_ORIE_MASK)

#define UART0_C3_TXINV_MASK          0x10u
#define UART0_C3_TXINV_SHIFT        4
#define UART0_C3_TXINV_WIDTH        1

#define UART0_C3_TXINV(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_TXINV_SHIFT))&UART0_C3_TXINV_MASK)

#define UART0_C3_TXDIR_MASK          0x20u
#define UART0_C3_TXDIR_SHIFT        5
#define UART0_C3_TXDIR_WIDTH        1

#define UART0_C3_TXDIR(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_TXDIR_SHIFT))&UART0_C3_TXDIR_MASK)

#define UART0_C3_R9T8_MASK          0x40u
#define UART0_C3_R9T8_SHIFT         6
#define UART0_C3_R9T8_WIDTH         1

#define UART0_C3_R9T8(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_R9T8_SHIFT))&UART0_C3_R9T8_MASK)

#define UART0_C3_R8T9_MASK          0x80u
#define UART0_C3_R8T9_SHIFT         7
#define UART0_C3_R8T9_WIDTH         1

#define UART0_C3_R8T9(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_R8T9_SHIFT))&UART0_C3_R8T9_MASK)

/* D Bit Fields */

#define UART0_D_ROT0_MASK           0x1u
#define UART0_D_ROT0_SHIFT          0
#define UART0_D_ROT0_WIDTH          1

#define UART0_D_ROT0(x)
(((uint8_t)((uint8_t)(x))<<UART0_D_ROT0_SHIFT))&UART0_D_ROT0_MASK)

#define UART0_D_R1T1_MASK           0x2u
#define UART0_D_R1T1_SHIFT          1
#define UART0_D_R1T1_WIDTH          1

```

```
#define UART0_D_R1T1(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R1T1_SHIFT))&UART0_D_R1T1_MASK)

#define UART0_D_R2T2_MASK          0x4u
#define UART0_D_R2T2_SHIFT         2
#define UART0_D_R2T2_WIDTH         1

#define UART0_D_R2T2(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R2T2_SHIFT))&UART0_D_R2T2_MASK)

#define UART0_D_R3T3_MASK          0x8u
#define UART0_D_R3T3_SHIFT         3
#define UART0_D_R3T3_WIDTH         1

#define UART0_D_R3T3(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R3T3_SHIFT))&UART0_D_R3T3_MASK)

#define UART0_D_R4T4_MASK          0x10u
#define UART0_D_R4T4_SHIFT         4
#define UART0_D_R4T4_WIDTH         1

#define UART0_D_R4T4(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R4T4_SHIFT))&UART0_D_R4T4_MASK)

#define UART0_D_R5T5_MASK          0x20u
#define UART0_D_R5T5_SHIFT         5
#define UART0_D_R5T5_WIDTH         1

#define UART0_D_R5T5(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R5T5_SHIFT))&UART0_D_R5T5_MASK)

#define UART0_D_R6T6_MASK          0x40u
#define UART0_D_R6T6_SHIFT         6
#define UART0_D_R6T6_WIDTH         1

#define UART0_D_R6T6(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R6T6_SHIFT))&UART0_D_R6T6_MASK)

#define UART0_D_R7T7_MASK          0x80u
#define UART0_D_R7T7_SHIFT         7
#define UART0_D_R7T7_WIDTH         1
```

```

#define UART0_D_R7T7(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R7T7_SHIFT))&UART0_D_R7T7_MASK)

/* MA1 Bit Fields */

#define UART0_MA1_MA_MASK          0xFFu
#define UART0_MA1_MA_SHIFT         0
#define UART0_MA1_MA_WIDTH         8

#define UART0_MA1_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA1_MA_SHIFT))&UART0_MA1_MA_MASK)

/* MA2 Bit Fields */

#define UART0_MA2_MA_MASK          0xFFu
#define UART0_MA2_MA_SHIFT         0
#define UART0_MA2_MA_WIDTH         8

#define UART0_MA2_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA2_MA_SHIFT))&UART0_MA2_MA_MASK)

/* C4 Bit Fields */

#define UART0_C4_OSR_MASK          0x1Fu
#define UART0_C4_OSR_SHIFT         0
#define UART0_C4_OSR_WIDTH         5

#define UART0_C4_OSR(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_OSR_SHIFT))&UART0_C4_OSR_MASK)

#define UART0_C4_M10_MASK          0x20u
#define UART0_C4_M10_SHIFT         5
#define UART0_C4_M10_WIDTH         1

#define UART0_C4_M10(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_M10_SHIFT))&UART0_C4_M10_MASK)

#define UART0_C4_MAEN2_MASK         0x40u
#define UART0_C4_MAEN2_SHIFT        6
#define UART0_C4_MAEN2_WIDTH        1

#define UART0_C4_MAEN2(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN2_SHIFT))&UART0_C4_MAEN2_MASK)

#define UART0_C4_MAEN1_MASK         0x80u

```

```

#define UART0_C4_MAEN1_SHIFT          7
#define UART0_C4_MAEN1_WIDTH          1
#define UART0_C4_MAEN1(x) (((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN1_SHIFT))&UART0_C4_MAEN1_MASK)

/* C5 Bit Fields */

#define UART0_C5_RESETDISC_MASK        0x1u
#define UART0_C5_RESETDISC_SHIFT       0
#define UART0_C5_RESETDISC_WIDTH      1
#define UART0_C5_RESETDISC(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RESETDISC_SHIFT))&UART0_C5_RESETDISC_MASK)

#define UART0_C5_BOTHEDGE_MASK         0x2u
#define UART0_C5_BOTHEDGE_SHIFT        1
#define UART0_C5_BOTHEDGE_WIDTH       1
#define UART0_C5_BOTHEDGE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_BOTHEDGE_SHIFT))&UART0_C5_BOTHEDGE_MASK)

#define UART0_C5_RDMAE_MASK            0x20u
#define UART0_C5_RDMAE_SHIFT           5
#define UART0_C5_RDMAE_WIDTH          1
#define UART0_C5_RDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RDMAE_SHIFT))&UART0_C5_RDMAE_MASK)

#define UART0_C5_TDMAE_MASK            0x80u
#define UART0_C5_TDMAE_SHIFT           7
#define UART0_C5_TDMAE_WIDTH          1
#define UART0_C5_TDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_TDMAE_SHIFT))&UART0_C5_TDMAE_MASK)

*/
* @}
/* end of group UART0_Register_Masks */

```

```
/* UART0 - Peripheral instance base addresses */

/** Peripheral UART0 base address */

#define UART0_BASE          (0x4006A000u)

/** Peripheral UART0 base pointer */

#define UART0           ((UART0_Type *)UART0_BASE)

#define UART0_BASE_PTR      (UART0)

/** Array initializer of UART0 peripheral base addresses */

#define UART0_BASE_ADDRS     { UART0_BASE }

/** Array initializer of UART0 peripheral base pointers */

#define UART0_BASE_PTRS      { UART0 }
```

```
/*
-----  
-- UART0 - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros  
 * @{  
 */
```

```
/* UART0 - Register instance definitions */

/* UART0 */

#define UART0_BDH          UART0_BDH_REG(UART0)  
  
#define UART0_BDL          UART0_BDL_REG(UART0)  
  
#define UART0_C1           UART0_C1_REG(UART0)  
  
#define UART0_C2           UART0_C2_REG(UART0)  
  
#define UART0_S1           UART0_S1_REG(UART0)
```

```
#define UART0_S2          UART0_S2_REG(UART0)
#define UART0_C3          UART0_C3_REG(UART0)
#define UART0_D           UART0_D_REG(UART0)
#define UART0_MA1         UART0_MA1_REG(UART0)
#define UART0_MA2         UART0_MA2_REG(UART0)
#define UART0_C4           UART0_C4_REG(UART0)
#define UART0_C5           UART0_C5_REG(UART0)
```

```
/*!
 * @}
 */
/* end of group UART0_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* end of group UART0_Peripheral_Access_Layer */
```

```
/* -----
-- USB Peripheral Access Layer
----- */
```

```
/*
 * @addtogroup USB_Peripheral_Access_Layer USB Peripheral Access Layer
 * @{
 */

```

```
/** USB - Register Layout Typedef */
```

```
typedef struct {
```

```
__I uint8_t PERID;           /**< Peripheral ID register, offset: 0x0 */

    uint8_t RESERVED_0[3];

__I uint8_t IDCOMP;          /**< Peripheral ID Complement register, offset: 0x4 */

    uint8_t RESERVED_1[3];

__I uint8_t REV;             /**< Peripheral Revision register, offset: 0x8 */

    uint8_t RESERVED_2[3];

__I uint8_t ADDINFO;         /**< Peripheral Additional Info register, offset: 0xC */

    uint8_t RESERVED_3[3];

__IO uint8_t OTGISTAT;       /**< OTG Interrupt Status register, offset: 0x10 */

    uint8_t RESERVED_4[3];

__IO uint8_t OTGICR;          /**< OTG Interrupt Control Register, offset: 0x14 */

    uint8_t RESERVED_5[3];

__IO uint8_t OTGSTAT;         /**< OTG Status register, offset: 0x18 */

    uint8_t RESERVED_6[3];

__IO uint8_t OTGCTL;          /**< OTG Control register, offset: 0x1C */

    uint8_t RESERVED_7[99];

__IO uint8_t ISTAT;           /**< Interrupt Status register, offset: 0x80 */

    uint8_t RESERVED_8[3];

__IO uint8_t INTEN;           /**< Interrupt Enable register, offset: 0x84 */

    uint8_t RESERVED_9[3];

__IO uint8_t ERRSTAT;         /**< Error Interrupt Status register, offset: 0x88 */

    uint8_t RESERVED_10[3];

__IO uint8_t ERREN;            /**< Error Interrupt Enable register, offset: 0x8C */

    uint8_t RESERVED_11[3];

__I uint8_t STAT;              /**< Status register, offset: 0x90 */

    uint8_t RESERVED_12[3];

__IO uint8_t CTL;               /**< Control register, offset: 0x94 */

    uint8_t RESERVED_13[3];

__IO uint8_t ADDR;              /**< Address register, offset: 0x98 */
```

```

    uint8_t RESERVED_14[3];

    __IO uint8_t BDTPAGE1;           /**< BDT Page Register 1, offset: 0x9C */

    uint8_t RESERVED_15[3];

    __IO uint8_t FRMNUML;          /**< Frame Number Register Low, offset: 0xA0 */

    uint8_t RESERVED_16[3];

    __IO uint8_t FRMNUMH;          /**< Frame Number Register High, offset: 0xA4 */

    uint8_t RESERVED_17[3];

    __IO uint8_t TOKEN;            /**< Token register, offset: 0xA8 */

    uint8_t RESERVED_18[3];

    __IO uint8_t SOFTHLD;          /**< SOF Threshold Register, offset: 0xAC */

    uint8_t RESERVED_19[3];

    __IO uint8_t BDTPAGE2;          /**< BDT Page Register 2, offset: 0xB0 */

    uint8_t RESERVED_20[3];

    __IO uint8_t BDTPAGE3;          /**< BDT Page Register 3, offset: 0xB4 */

    uint8_t RESERVED_21[11];

struct {                           /* offset: 0xC0, array step: 0x4 */

    __IO uint8_t ENDPT;           /**< Endpoint Control register, array offset: 0xC0, array step:
        0x4 */

    uint8_t RESERVED_0[3];

} ENDPOINT[16];

    __IO uint8_t USBCTRL;          /**< USB Control register, offset: 0x100 */

    uint8_t RESERVED_22[3];

    __I uint8_t OBSERVE;          /**< USB OTG Observe register, offset: 0x104 */

    uint8_t RESERVED_23[3];

    __IO uint8_t CONTROL;          /**< USB OTG Control register, offset: 0x108 */

    uint8_t RESERVED_24[3];

    __IO uint8_t USBTRCO;          /**< USB Transceiver Control Register 0, offset: 0x10C */

    uint8_t RESERVED_25[7];

    __IO uint8_t USBFRMADJUST;     /**< Frame Adjust Register, offset: 0x114 */

```

```

} USB_Type, *USB_MemMapPtr;

/*
-- USB - Register accessor macros
----- */
/*!
* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros
* @{
*/
/* USB - Register accessors */
#define USB_PERID_REG(base) ((base)->PERID)
#define USB_IDCOMP_REG(base) ((base)->IDCOMP)
#define USB_REV_REG(base) ((base)->REV)
#define USB_ADDINFO_REG(base) ((base)->ADDINFO)
#define USB_OTGISTAT_REG(base) ((base)->OTGISTAT)
#define USB_OTGICR_REG(base) ((base)->OTGICR)
#define USB_OTGSTAT_REG(base) ((base)->OTGSTAT)
#define USB_OTGCTL_REG(base) ((base)->OTGCTL)
#define USB_ISTAT_REG(base) ((base)->ISTAT)
#define USB_INTEN_REG(base) ((base)->INTEN)
#define USB_ERRSTAT_REG(base) ((base)->ERRSTAT)
#define USB_ERREN_REG(base) ((base)->ERREN)
#define USB_STAT_REG(base) ((base)->STAT)
#define USB_CTL_REG(base) ((base)->CTL)
#define USB_ADDR_REG(base) ((base)->ADDR)
#define USB_BDTPAGE1_REG(base) ((base)->BDTPAGE1)

```

```
#define USB_FRMNUML_REG(base)          ((base)->FRMNUML)
#define USB_FRMNUMH_REG(base)          ((base)->FRMNUMH)
#define USB_TOKEN_REG(base)           ((base)->TOKEN)
#define USB_SOFTHLD_REG(base)          ((base)->SOFTHLD)
#define USB_BDTPAGE2_REG(base)          ((base)->BDTPAGE2)
#define USB_BDTPAGE3_REG(base)          ((base)->BDTPAGE3)
#define USB_ENDPT_REG(base,index)      ((base)->ENDPOINT[index].ENDPT)
#define USB_ENDPT_COUNT                16
#define USB_USBCTRL_REG(base)          ((base)->USBCTRL)
#define USB_OBSERVE_REG(base)          ((base)->OBSERVE)
#define USB_CONTROL_REG(base)          ((base)->CONTROL)
#define USB_USBTRCO_REG(base)          ((base)->USBTRCO)
#define USB_USBFRMADJUST_REG(base)     ((base)->USBFRMADJUST)
```

```
/*!
 * @}
 */
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*
-----  
-- USB Register Masks  
----- */
```

```
/*!  
 * @addtogroup USB_Register_Masks USB Register Masks  
 * @{  
 */
```

```
/* PERID Bit Fields */
```

```

#define USB_PERID_ID_MASK          0x3Fu
#define USB_PERID_ID_SHIFT         0
#define USB_PERID_ID_WIDTH         6
#define USB_PERID_ID(x) (((uint8_t)((uint8_t)(x))<<USB_PERID_ID_SHIFT))&USB_PERID_ID_MASK

/* IDCOMP Bit Fields */

#define USB_IDCOMP_NID_MASK        0x3Fu
#define USB_IDCOMP_NID_SHIFT       0
#define USB_IDCOMP_NID_WIDTH       6
#define USB_IDCOMP_NID(x) (((uint8_t)((uint8_t)(x))<<USB_IDCOMP_NID_SHIFT))&USB_IDCOMP_NID_MASK

/* REV Bit Fields */

#define USB_REV_REV_MASK           0xFFu
#define USB_REV_REV_SHIFT          0
#define USB_REV_REV_WIDTH          8
#define USB_REV_REV(x) (((uint8_t)((uint8_t)(x))<<USB_REV_REV_SHIFT))&USB_REV_REV_MASK

/* ADDINFO Bit Fields */

#define USB_ADDINFO_IHOST_MASK     0x1u
#define USB_ADDINFO_IHOST_SHIFT    0
#define USB_ADDINFO_IHOST_WIDTH    1
#define USB_ADDINFO_IHOST(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IHOST_SHIFT))&USB_ADDINFO_IHOST_MASK

#define USB_ADDINFO_IRQNUM_MASK    0xF8u
#define USB_ADDINFO_IRQNUM_SHIFT   3
#define USB_ADDINFO_IRQNUM_WIDTH   5
#define USB_ADDINFO_IRQNUM(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IRQNUM_SHIFT))&USB_ADDINFO_IRQNUM_MASK

/* OTGISTAT Bit Fields */

#define USB_OTGISTAT_AVBUSCHG_MASK 0x1u

```

```

#define USB_OTGISTAT_AVBUSCHG_SHIFT      0
#define USB_OTGISTAT_AVBUSCHG_WIDTH      1
#define USB_OTGISTAT_AVBUSCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_AVBUSCHG_SHIFT))&USB_OTGISTAT_AVBUSCHG_MASK)

#define USB_OTGISTAT_B_SESS_CHG_MASK      0x4u
#define USB_OTGISTAT_B_SESS_CHG_SHIFT     2
#define USB_OTGISTAT_B_SESS_CHG_WIDTH    1
#define USB_OTGISTAT_B_SESS_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_B_SESS_CHG_SHIFT))&USB_OTGISTAT_B_SESS_CHG_MASK)

#define USB_OTGISTAT_SESSVLDCHG_MASK      0x8u
#define USB_OTGISTAT_SESSVLDCHG_SHIFT     3
#define USB_OTGISTAT_SESSVLDCHG_WIDTH    1
#define USB_OTGISTAT_SESSVLDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_SESSVLDCHG_SHIFT))&USB_OTGISTAT_SESSVLDCHG_MASK)

#define USB_OTGISTAT_LINE_STATE_CHG_MASK   0x20u
#define USB_OTGISTAT_LINE_STATE_CHG_SHIFT  5
#define USB_OTGISTAT_LINE_STATE_CHG_WIDTH  1
#define USB_OTGISTAT_LINE_STATE_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_LINE_STATE_CHG_SHIFT))&USB_OTGISTAT_LINE_STATE_CHG_MASK)

#define USB_OTGISTAT_ONEMSEC_MASK         0x40u
#define USB_OTGISTAT_ONEMSEC_SHIFT       6
#define USB_OTGISTAT_ONEMSEC_WIDTH      1
#define USB_OTGISTAT_ONEMSEC(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_ONEMSEC_SHIFT))&USB_OTGISTAT_ONEMSEC_MASK)

#define USB_OTGISTAT_IDCHG_MASK          0x80u
#define USB_OTGISTAT_IDCHG_SHIFT        7
#define USB_OTGISTAT_IDCHG_WIDTH       1
#define USB_OTGISTAT_IDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_IDCHG_SHIFT))&USB_OTGISTAT_IDCHG_MASK)

/* OTGICR Bit Fields */

```

```

#define USB_OTGICR_AVBUSEN_MASK      0x1u
#define USB_OTGICR_AVBUSEN_SHIFT     0
#define USB_OTGICR_AVBUSEN_WIDTH     1
#define USB_OTGICR_AVBUSEN(x)        (((uint8_t)((uint8_t)(x))<<USB_OTGICR_AVBUSEN_SHIFT))&USB_OTGICR_AVBUSEN_MASK)
#define USB_OTGICR_BSESSEN_MASK      0x4u
#define USB_OTGICR_BSESSEN_SHIFT     2
#define USB_OTGICR_BSESSEN_WIDTH     1
#define USB_OTGICR_BSESSEN(x)        (((uint8_t)((uint8_t)(x))<<USB_OTGICR_BSESSEN_SHIFT))&USB_OTGICR_BSESSEN_MASK)
#define USB_OTGICR_SESSVLDEN_MASK    0x8u
#define USB_OTGICR_SESSVLDEN_SHIFT   3
#define USB_OTGICR_SESSVLDEN_WIDTH   1
#define USB_OTGICR_SESSVLDEN(x)      (((uint8_t)((uint8_t)(x))<<USB_OTGICR_SESSVLDEN_SHIFT))&USB_OTGICR_SESSVLDEN_MASK)
#define USB_OTGICR_LINESTATEEN_MASK   0x20u
#define USB_OTGICR_LINESTATEEN_SHIFT  5
#define USB_OTGICR_LINESTATEEN_WIDTH  1
#define USB_OTGICR_LINESTATEEN(x)     (((uint8_t)((uint8_t)(x))<<USB_OTGICR_LINESTATEEN_SHIFT))&USB_OTGICR_LINESTATEEN_MASK)
#define USB_OTGICR_ONEMSECEN_MASK    0x40u
#define USB_OTGICR_ONEMSECEN_SHIFT   6
#define USB_OTGICR_ONEMSECEN_WIDTH   1
#define USB_OTGICR_ONEMSECEN(x)      (((uint8_t)((uint8_t)(x))<<USB_OTGICR_ONEMSECEN_SHIFT))&USB_OTGICR_ONEMSECEN_MASK)
#define USB_OTGICR_IDEN_MASK         0x80u
#define USB_OTGICR_IDEN_SHIFT        7
#define USB_OTGICR_IDEN_WIDTH        1
#define USB_OTGICR_IDEN(x)          (((uint8_t)((uint8_t)(x))<<USB_OTGICR_IDEN_SHIFT))&USB_OTGICR_IDEN_MASK)
/* OTGSTAT Bit Fields */

```

```

#define USB_OTGSTAT_AVBUSVLD_MASK      0x1u
#define USB_OTGSTAT_AVBUSVLD_SHIFT     0
#define USB_OTGSTAT_AVBUSVLD_WIDTH     1
#define USB_OTGSTAT_AVBUSVLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_AVBUSVLD_SHIFT))&USB_OTGSTAT_AVBUSVLD_MASK

#define USB_OTGSTAT_BSESSEND_MASK       0x4u
#define USB_OTGSTAT_BSESSEND_SHIFT     2
#define USB_OTGSTAT_BSESSEND_WIDTH     1
#define USB_OTGSTAT_BSESSEND(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_BSESSEND_SHIFT))&USB_OTGSTAT_BSESSEND_MASK

#define USB_OTGSTAT_SESS_VLD_MASK       0x8u
#define USB_OTGSTAT_SESS_VLD_SHIFT     3
#define USB_OTGSTAT_SESS_VLD_WIDTH     1
#define USB_OTGSTAT_SESS_VLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_SESS_VLD_SHIFT))&USB_OTGSTAT_SESS_VLD_MASK

#define USB_OTGSTAT_LINESTATESTABLE_MASK 0x20u
#define USB_OTGSTAT_LINESTATESTABLE_SHIFT 5
#define USB_OTGSTAT_LINESTATESTABLE_WIDTH 1
#define USB_OTGSTAT_LINESTATESTABLE(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_LINESTATESTABLE_SHIFT))&USB_OTGSTAT_LINESTATESTABLE_MASK

#define USB_OTGSTAT_ONEMSECEN_MASK      0x40u
#define USB_OTGSTAT_ONEMSECEN_SHIFT     6
#define USB_OTGSTAT_ONEMSECEN_WIDTH     1
#define USB_OTGSTAT_ONEMSECEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ONEMSECEN_SHIFT))&USB_OTGSTAT_ONEMSECEN_MASK

#define USB_OTGSTAT_ID_MASK            0x80u
#define USB_OTGSTAT_ID_SHIFT          7
#define USB_OTGSTAT_ID_WIDTH          1
#define USB_OTGSTAT_ID(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ID_SHIFT))&USB_OTGSTAT_ID_MASK

```

```

/* OTGCTL Bit Fields */

#define USB_OTGCTL_OTGEN_MASK          0x4u
#define USB_OTGCTL_OTGEN_SHIFT         2
#define USB_OTGCTL_OTGEN_WIDTH         1
#define USB_OTGCTL_OTGEN(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_OTGEN_SHIFT))&USB_OTGCTL_OTGEN_MASK)

#define USB_OTGCTL_DMLOW_MASK          0x10u
#define USB_OTGCTL_DMLOW_SHIFT         4
#define USB_OTGCTL_DMLOW_WIDTH         1
#define USB_OTGCTL_DMLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DMLOW_SHIFT))&USB_OTGCTL_DMLOW_MASK)

#define USB_OTGCTL_DPLOW_MASK          0x20u
#define USB_OTGCTL_DPLOW_SHIFT         5
#define USB_OTGCTL_DPLOW_WIDTH         1
#define USB_OTGCTL_DPLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPLOW_SHIFT))&USB_OTGCTL_DPLOW_MASK)

#define USB_OTGCTL_DPHIGH_MASK         0x80u
#define USB_OTGCTL_DPHIGH_SHIFT        7
#define USB_OTGCTL_DPHIGH_WIDTH        1
#define USB_OTGCTL_DPHIGH(x)           (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPHIGH_SHIFT))&USB_OTGCTL_DPHIGH_MASK)

/* ISTAT Bit Fields */

#define USB_ISTAT_USBRST_MASK          0x1u
#define USB_ISTAT_USBRST_SHIFT         0
#define USB_ISTAT_USBRST_WIDTH         1
#define USB_ISTAT_USBRST(x)            (((uint8_t)((uint8_t)(x)<<USB_ISTAT_USBRST_SHIFT))&USB_ISTAT_USBRST_MASK)

#define USB_ISTAT_ERROR_MASK           0x2u
#define USB_ISTAT_ERROR_SHIFT          1
#define USB_ISTAT_ERROR_WIDTH          1

```

```

#define USB_ISTAT_ERROR(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ERROR_SHIFT))&USB_ISTAT_ERROR_MASK)

#define USB_ISTAT_SOFTOK_MASK          0x4u
#define USB_ISTAT_SOFTOK_SHIFT        2
#define USB_ISTAT_SOFTOK_WIDTH        1

#define USB_ISTAT_SOFTOK(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SOFTOK_SHIFT))&USB_ISTAT_SOFTOK_MASK)

#define USB_ISTAT_TOKDNE_MASK         0x8u
#define USB_ISTAT_TOKDNE_SHIFT        3
#define USB_ISTAT_TOKDNE_WIDTH        1

#define USB_ISTAT_TOKDNE(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_TOKDNE_SHIFT))&USB_ISTAT_TOKDNE_MASK)

#define USB_ISTAT_SLEEP_MASK          0x10u
#define USB_ISTAT_SLEEP_SHIFT         4
#define USB_ISTAT_SLEEP_WIDTH         1

#define USB_ISTAT_SLEEP(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SLEEP_SHIFT))&USB_ISTAT_SLEEP_MASK)

#define USB_ISTAT_RESUME_MASK         0x20u
#define USB_ISTAT_RESUME_SHIFT        5
#define USB_ISTAT_RESUME_WIDTH        1

#define USB_ISTAT_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_RESUME_SHIFT))&USB_ISTAT_RESUME_MASK)

#define USB_ISTAT_ATTACH_MASK          0x40u
#define USB_ISTAT_ATTACH_SHIFT         6
#define USB_ISTAT_ATTACH_WIDTH         1

#define USB_ISTAT_ATTACH(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ATTACH_SHIFT))&USB_ISTAT_ATTACH_MASK)

#define USB_ISTAT_STALL_MASK          0x80u
#define USB_ISTAT_STALL_SHIFT         7
#define USB_ISTAT_STALL_WIDTH         1

```

```

#define USB_ISTAT_STALL(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_STALL_SHIFT))&USB_ISTAT_STALL_MASK)

/* INTEN Bit Fields */

#define USB_INTEN_USBRSTEN_MASK          0x1u
#define USB_INTEN_USBRSTEN_SHIFT         0
#define USB_INTEN_USBRSTEN_WIDTH         1
#define USB_INTEN_USBRSTEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_USBRSTEN_SHIFT))&USB_INTEN_USBRSTEN_MASK)

#define USB_INTEN_ERROREN_MASK           0x2u
#define USB_INTEN_ERROREN_SHIFT          1
#define USB_INTEN_ERROREN_WIDTH          1
#define USB_INTEN_ERROREN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ERROREN_SHIFT))&USB_INTEN_ERROREN_MASK)

#define USB_INTEN_SOFTOKEN_MASK          0x4u
#define USB_INTEN_SOFTOKEN_SHIFT         2
#define USB_INTEN_SOFTOKEN_WIDTH         1
#define USB_INTEN_SOFTOKEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SOFTOKEN_SHIFT))&USB_INTEN_SOFTOKEN_MASK)

#define USB_INTEN_TOKDNEEN_MASK          0x8u
#define USB_INTEN_TOKDNEEN_SHIFT         3
#define USB_INTEN_TOKDNEEN_WIDTH         1
#define USB_INTEN_TOKDNEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_TOKDNEEN_SHIFT))&USB_INTEN_TOKDNEEN_MASK)

#define USB_INTEN_SLEEPEN_MASK           0x10u
#define USB_INTEN_SLEEPEN_SHIFT          4
#define USB_INTEN_SLEEPEN_WIDTH          1
#define USB_INTEN_SLEEPEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SLEEPEN_SHIFT))&USB_INTEN_SLEEPEN_MASK)

#define USB_INTEN_RESUMEEN_MASK          0x20u
#define USB_INTEN_RESUMEEN_SHIFT         5
#define USB_INTEN_RESUMEEN_WIDTH         1

```

```

#define USB_INTEN_RESUMEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_RESUMEEN_SHIFT))&USB_INTEN_RESUMEEN_MASK)

#define USB_INTEN_ATTACHEN_MASK          0x40u
#define USB_INTEN_ATTACHEN_SHIFT        6
#define USB_INTEN_ATTACHEN_WIDTH        1

#define USB_INTEN_ATTACHEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ATTACHEN_SHIFT))&USB_INTEN_ATTACHEN_MASK)

#define USB_INTEN_STALLEN_MASK          0x80u
#define USB_INTEN_STALLEN_SHIFT        7
#define USB_INTEN_STALLEN_WIDTH        1

#define USB_INTEN_STALLEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_STALLEN_SHIFT))&USB_INTEN_STALLEN_MASK)

/* ERRSTAT Bit Fields */

#define USB_ERRSTAT_PIDERR_MASK         0x1u
#define USB_ERRSTAT_PIDERR_SHIFT       0
#define USB_ERRSTAT_PIDERR_WIDTH       1

#define USB_ERRSTAT_PIDERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_PIDERR_SHIFT))&USB_ERRSTAT_PIDERR_MASK)

#define USB_ERRSTAT_CRC5EOF_MASK         0x2u
#define USB_ERRSTAT_CRC5EOF_SHIFT       1
#define USB_ERRSTAT_CRC5EOF_WIDTH       1

#define USB_ERRSTAT_CRC5EOF(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC5EOF_SHIFT))&USB_ERRSTAT_CRC5EOF_MASK)

#define USB_ERRSTAT_CRC16_MASK          0x4u
#define USB_ERRSTAT_CRC16_SHIFT        2
#define USB_ERRSTAT_CRC16_WIDTH        1

#define USB_ERRSTAT_CRC16(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC16_SHIFT))&USB_ERRSTAT_CRC16_MASK)

#define USB_ERRSTAT_DFN8_MASK           0x8u
#define USB_ERRSTAT_DFN8_SHIFT         3
#define USB_ERRSTAT_DFN8_WIDTH         1

```

```

#define USB_ERRSTAT_DFN8(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DFN8_SHIFT))&USB_ERRSTAT_DFN8_MASK)

#define USB_ERRSTAT_BTOERR_MASK          0x10u
#define USB_ERRSTAT_BTOERR_SHIFT        4
#define USB_ERRSTAT_BTOERR_WIDTH        1

#define USB_ERRSTAT_BTOERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTOERR_SHIFT))&USB_ERRSTAT_BTOERR_MASK)

#define USB_ERRSTAT_DMAERR_MASK         0x20u
#define USB_ERRSTAT_DMAERR_SHIFT       5
#define USB_ERRSTAT_DMAERR_WIDTH       1

#define USB_ERRSTAT_DMAERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DMAERR_SHIFT))&USB_ERRSTAT_DMAERR_MASK)

#define USB_ERRSTAT_BTSERR_MASK         0x80u
#define USB_ERRSTAT_BTSERR_SHIFT       7
#define USB_ERRSTAT_BTSERR_WIDTH       1

#define USB_ERRSTAT_BTSERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTSERR_SHIFT))&USB_ERRSTAT_BTSERR_MASK)

/* ERREN Bit Fields */

#define USB_ERREN_PIDERREN_MASK         0x1u
#define USB_ERREN_PIDERREN_SHIFT        0
#define USB_ERREN_PIDERREN_WIDTH        1

#define USB_ERREN_PIDERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_PIDERREN_SHIFT))&USB_ERREN_PIDERREN_MASK)

#define USB_ERREN_CRC5EOFEN_MASK        0x2u
#define USB_ERREN_CRC5EOFEN_SHIFT      1
#define USB_ERREN_CRC5EOFEN_WIDTH      1

#define USB_ERREN_CRC5EOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_CRC5EOFEN_SHIFT))&USB_ERREN_CRC5EOFEN_MASK)

#define USB_ERREN_CRC16EN_MASK          0x4u
#define USB_ERREN_CRC16EN_SHIFT        2
#define USB_ERREN_CRC16EN_WIDTH        1

```

```

#define USB_ERREN_CRC16EN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_CRC16EN_SHIFT))&USB_ERREN_CRC16EN_MASK)

#define USB_ERREN_DFN8EN_MASK          0x8u
#define USB_ERREN_DFN8EN_SHIFT         3
#define USB_ERREN_DFN8EN_WIDTH         1

#define USB_ERREN_DFN8EN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_DFN8EN_SHIFT))&USB_ERREN_DFN8EN_MASK)

#define USB_ERREN_BTOERREN_MASK        0x10u
#define USB_ERREN_BTOERREN_SHIFT       4
#define USB_ERREN_BTOERREN_WIDTH       1

#define USB_ERREN_BTOERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_BTOERREN_SHIFT))&USB_ERREN_BTOERREN_MASK)

#define USB_ERREN_DMAERREN_MASK        0x20u
#define USB_ERREN_DMAERREN_SHIFT       5
#define USB_ERREN_DMAERREN_WIDTH       1

#define USB_ERREN_DMAERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_DMAERREN_SHIFT))&USB_ERREN_DMAERREN_MASK)

#define USB_ERREN_BTSERREN_MASK        0x80u
#define USB_ERREN_BTSERREN_SHIFT       7
#define USB_ERREN_BTSERREN_WIDTH       1

#define USB_ERREN_BTSERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_BTSERREN_SHIFT))&USB_ERREN_BTSERREN_MASK)

/* STAT Bit Fields */

#define USB_STAT_ODD_MASK             0x4u
#define USB_STAT_ODD_SHIFT             2
#define USB_STAT_ODD_WIDTH              1

#define USB_STAT_ODD(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_ODD_SHIFT))&USB_STAT_ODD_MASK)

#define USB_STAT_TX_MASK               0x8u
#define USB_STAT_TX_SHIFT              3
#define USB_STAT_TX_WIDTH              1

```

```

#define USB_STAT_TX(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_TX_SHIFT))&USB_STAT_TX_MASK)

#define USB_STAT_ENDP_MASK          0xF0u
#define USB_STAT_ENDP_SHIFT        4
#define USB_STAT_ENDP_WIDTH        4

#define USB_STAT_ENDP(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_ENDP_SHIFT))&USB_STAT_ENDP_MASK)

/* CTL Bit Fields */

#define USB_CTL_USBENSOFEN_MASK      0x1u
#define USB_CTL_USBENSOFEN_SHIFT    0
#define USB_CTL_USBENSOFEN_WIDTH    1

#define USB_CTL_USBENSOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_USBENSOFEN_SHIFT))&USB_CTL_USBENSOFEN_MASK)

#define USB_CTL_ODDRST_MASK         0x2u
#define USB_CTL_ODDRST_SHIFT       1
#define USB_CTL_ODDRST_WIDTH       1

#define USB_CTL_ODDRST(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_ODDRST_SHIFT))&USB_CTL_ODDRST_MASK)

#define USB_CTL_RESUME_MASK         0x4u
#define USB_CTL_RESUME_SHIFT       2
#define USB_CTL_RESUME_WIDTH       1

#define USB_CTL_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESUME_SHIFT))&USB_CTL_RESUME_MASK)

#define USB_CTL_HOSTMODEEN_MASK     0x8u
#define USB_CTL_HOSTMODEEN_SHIFT   3
#define USB_CTL_HOSTMODEEN_WIDTH   1

#define USB_CTL_HOSTMODEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_HOSTMODEEN_SHIFT))&USB_CTL_HOSTMODEEN_MASK)

#define USB_CTL_RESET_MASK          0x10u
#define USB_CTL_RESET_SHIFT        4
#define USB_CTL_RESET_WIDTH        1

```

```

#define USB_CTL_RESET(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESET_SHIFT))&USB_CTL_RESET_MASK)

#define USB_CTL_TXSUSPENDTOKENBUSY_MASK      0x20u
#define USB_CTL_TXSUSPENDTOKENBUSY_SHIFT     5
#define USB_CTL_TXSUSPENDTOKENBUSY_WIDTH     1

#define USB_CTL_TXSUSPENDTOKENBUSY(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_TXSUSPENDTOKENBUSY_SHIFT))&USB_CTL_TXSUSPENDTOKENBUSY
_MASK)

#define USB_CTL_SE0_MASK                     0x40u
#define USB_CTL_SE0_SHIFT                   6
#define USB_CTL_SE0_WIDTH                  1

#define USB_CTL_SE0(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_SE0_SHIFT))&USB_CTL_SE0_MASK)

#define USB_CTL_JSTATE_MASK                 0x80u
#define USB_CTL_JSTATE_SHIFT                7
#define USB_CTL_JSTATE_WIDTH               1

#define USB_CTL_JSTATE(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_JSTATE_SHIFT))&USB_CTL_JSTATE_MASK)

/* ADDR Bit Fields */

#define USB_ADDR_ADDR_MASK                 0x7Fu
#define USB_ADDR_ADDR_SHIFT                0
#define USB_ADDR_ADDR_WIDTH               7

#define USB_ADDR_ADDR(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_ADDR_SHIFT))&USB_ADDR_ADDR_MASK)

#define USB_ADDR_LSEN_MASK                 0x80u
#define USB_ADDR_LSEN_SHIFT                7
#define USB_ADDR_LSEN_WIDTH               1

#define USB_ADDR_LSEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_LSEN_SHIFT))&USB_ADDR_LSEN_MASK)

/* BDTPAGE1 Bit Fields */

#define USB_BDTPAGE1_BDTBA_MASK           0xFEu

```

```

#define USB_BDTPAGE1_BDTBA_SHIFT      1
#define USB_BDTPAGE1_BDTBA_WIDTH      7
#define USB_BDTPAGE1_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE1_BDTBA_SHIFT))&USB_BDTPAGE1_BDTBA_MASK)

/* FRMNUML Bit Fields */

#define USB_FRMNUML_FRM_MASK          0xFFu
#define USB_FRMNUML_FRM_SHIFT         0
#define USB_FRMNUML_FRM_WIDTH         8
#define USB_FRMNUML_FRM(x) (((uint8_t)((uint8_t)(x)<<USB_FRMNUML_FRM_SHIFT))&USB_FRMNUML_FRM_MASK)

/* FRMNUMH Bit Fields */

#define USB_FRMNUMH_FRM_MASK          0x7u
#define USB_FRMNUMH_FRM_SHIFT         0
#define USB_FRMNUMH_FRM_WIDTH         3
#define USB_FRMNUMH_FRM(x) (((uint8_t)((uint8_t)(x)<<USB_FRMNUMH_FRM_SHIFT))&USB_FRMNUMH_FRM_MASK)

/* TOKEN Bit Fields */

#define USB_TOKEN_TOKENENDPT_MASK      0xFu
#define USB_TOKEN_TOKENENDPT_SHIFT     0
#define USB_TOKEN_TOKENENDPT_WIDTH     4
#define USB_TOKEN_TOKENENDPT(x) (((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENENDPT_SHIFT))&USB_TOKEN_TOKENENDPT_MASK)

#define USB_TOKEN_TOKENPID_MASK         0xF0u
#define USB_TOKEN_TOKENPID_SHIFT        4
#define USB_TOKEN_TOKENPID_WIDTH        4
#define USB_TOKEN_TOKENPID(x) (((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENPID_SHIFT))&USB_TOKEN_TOKENPID_MASK)

/* SOFTHLD Bit Fields */

#define USB_SOFTHLD_CNT_MASK          0xFFu
#define USB_SOFTHLD_CNT_SHIFT         0

```

```

#define USB_SOFTHLD_CNT_WIDTH          8
#define USB_SOFTHLD_CNT(x) (((uint8_t)((uint8_t)(x)<<USB_SOFTHLD_CNT_SHIFT))&USB_SOFTHLD_CNT_MASK)

/* BDTPAGE2 Bit Fields */

#define USB_BDTPAGE2_BDTBA_MASK        0xFFu
#define USB_BDTPAGE2_BDTBA_SHIFT       0
#define USB_BDTPAGE2_BDTBA_WIDTH       8
#define USB_BDTPAGE2_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE2_BDTBA_SHIFT))&USB_BDTPAGE2_BDTBA_MASK)

/* BDTPAGE3 Bit Fields */

#define USB_BDTPAGE3_BDTBA_MASK        0xFFu
#define USB_BDTPAGE3_BDTBA_SHIFT       0
#define USB_BDTPAGE3_BDTBA_WIDTH       8
#define USB_BDTPAGE3_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE3_BDTBA_SHIFT))&USB_BDTPAGE3_BDTBA_MASK)

/* ENDPT Bit Fields */

#define USB_ENDPT_EPHSHK_MASK          0x1u
#define USB_ENDPT_EPHSHK_SHIFT         0
#define USB_ENDPT_EPHSHK_WIDTH         1
#define USB_ENDPT_EPHSHK(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPHSHK_SHIFT))&USB_ENDPT_EPHSHK_MASK)

#define USB_ENDPT_EPSTALL_MASK         0x2u
#define USB_ENDPT_EPSTALL_SHIFT        1
#define USB_ENDPT_EPSTALL_WIDTH        1
#define USB_ENDPT_EPSTALL(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPSTALL_SHIFT))&USB_ENDPT_EPSTALL_MASK)

#define USB_ENDPT_EPTXEN_MASK          0x4u
#define USB_ENDPT_EPTXEN_SHIFT         2
#define USB_ENDPT_EPTXEN_WIDTH         1
#define USB_ENDPT_EPTXEN(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPTXEN_SHIFT))&USB_ENDPT_EPTXEN_MASK)

```

```

#define USB_ENDPT_EPRXEN_MASK          0x8u
#define USB_ENDPT_EPRXEN_SHIFT         3
#define USB_ENDPT_EPRXEN_WIDTH         1
#define USB_ENDPT_EPRXEN(x)            (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPRXEN_SHIFT))&USB_ENDPT_EPRXEN_MASK)

#define USB_ENDPT_EPCTLDIS_MASK        0x10u
#define USB_ENDPT_EPCTLDIS_SHIFT       4
#define USB_ENDPT_EPCTLDIS_WIDTH       1
#define USB_ENDPT_EPCTLDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPCTLDIS_SHIFT))&USB_ENDPT_EPCTLDIS_MASK)

#define USB_ENDPT_RETRYDIS_MASK        0x40u
#define USB_ENDPT_RETRYDIS_SHIFT       6
#define USB_ENDPT_RETRYDIS_WIDTH       1
#define USB_ENDPT_RETRYDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_RETRYDIS_SHIFT))&USB_ENDPT_RETRYDIS_MASK)

#define USB_ENDPT_HOSTWOHUB_MASK        0x80u
#define USB_ENDPT_HOSTWOHUB_SHIFT       7
#define USB_ENDPT_HOSTWOHUB_WIDTH       1
#define USB_ENDPT_HOSTWOHUB(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_HOSTWOHUB_SHIFT))&USB_ENDPT_HOSTWOHUB_MASK)

/* USBCTRL Bit Fields */

#define USB_USBCTRL_PDE_MASK           0x40u
#define USB_USBCTRL_PDE_SHIFT          6
#define USB_USBCTRL_PDE_WIDTH          1
#define USB_USBCTRL_PDE(x)             (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_PDE_SHIFT))&USB_USBCTRL_PDE_MASK)

#define USB_USBCTRL_SUSP_MASK           0x80u
#define USB_USBCTRL_SUSP_SHIFT          7
#define USB_USBCTRL_SUSP_WIDTH          1
#define USB_USBCTRL_SUSP(x)             (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_SUSP_SHIFT))&USB_USBCTRL_SUSP_MASK)

```

```

/* OBSERVE Bit Fields */

#define USB_OBSERVE_DMPD_MASK          0x10u
#define USB_OBSERVE_DMPD_SHIFT         4
#define USB_OBSERVE_DMPD_WIDTH         1
#define USB_OBSERVE_DMPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DMPD_SHIFT))&USB_OBSERVE_DMPD_MASK

#define USB_OBSERVE_DPPD_MASK          0x40u
#define USB_OBSERVE_DPPD_SHIFT         6
#define USB_OBSERVE_DPPD_WIDTH         1
#define USB_OBSERVE_DPPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPD_SHIFT))&USB_OBSERVE_DPPD_MASK

#define USB_OBSERVE_DPPU_MASK          0x80u
#define USB_OBSERVE_DPPU_SHIFT         7
#define USB_OBSERVE_DPPU_WIDTH         1
#define USB_OBSERVE_DPPU(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPU_SHIFT))&USB_OBSERVE_DPPU_MASK

/* CONTROL Bit Fields */

#define USB_CONTROL_DPPULLUPNONOTG_MASK 0x10u
#define USB_CONTROL_DPPULLUPNONOTG_SHIFT 4
#define USB_CONTROL_DPPULLUPNONOTG_WIDTH 1
#define USB_CONTROL_DPPULLUPNONOTG(x)   (((uint8_t)((uint8_t)(x))<<USB_CONTROL_DPPULLUPNONOTG_SHIFT))&USB_CONTROL_DPPULLUPNONOTG_MASK

/* USBTRCO Bit Fields */

#define USB_USBTRCO_USB_RESUME_INT_MASK 0x1u
#define USB_USBTRCO_USB_RESUME_INT_SHIFT 0
#define USB_USBTRCO_USB_RESUME_INT_WIDTH 1
#define USB_USBTRCO_USB_RESUME_INT(x)   (((uint8_t)((uint8_t)(x))<<USB_USBTRCO_USB_RESUME_INT_SHIFT))&USB_USBTRCO_USB_RESUME_INT_MASK

#define USB_USBTRCO_SYNC_DET_MASK      0x2u

```

```

#define USB_USBTRCO_SYNC_DET_SHIFT      1
#define USB_USBTRCO_SYNC_DET_WIDTH     1
#define USB_USBTRCO_SYNC_DET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_SYNC_DET_SHIFT))&USB_USBTRCO_SYNC_DET_MASK)

#define USB_USBTRCO_USBRESMEN_MASK      0x20u
#define USB_USBTRCO_USBRESMEN_SHIFT     5
#define USB_USBTRCO_USBRESMEN_WIDTH     1
#define USB_USBTRCO_USBRESMEN(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESMEN_SHIFT))&USB_USBTRCO_USBRESMEN_MASK)

#define USB_USBTRCO_USBRESET_MASK       0x80u
#define USB_USBTRCO_USBRESET_SHIFT      7
#define USB_USBTRCO_USBRESET_WIDTH      1
#define USB_USBTRCO_USBRESET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESET_SHIFT))&USB_USBTRCO_USBRESET_MASK)

/* USBFRMADJUST Bit Fields */

#define USB_USBFRMADJUST_ADJ_MASK       0xFFu
#define USB_USBFRMADJUST_ADJ_SHIFT      0
#define USB_USBFRMADJUST_ADJ_WIDTH      8
#define USB_USBFRMADJUST_ADJ(x)
(((uint8_t)((uint8_t)(x)<<USB_USBFRMADJUST_ADJ_SHIFT))&USB_USBFRMADJUST_ADJ_MASK)

/*
 * @}
 */

/* end of group USB_Register_Masks */

```

```

/* USB - Peripheral instance base addresses */
/** Peripheral USBO base address */
#define USBO_BASE                      (0x40072000u)
/** Peripheral USBO base pointer */

```

```

#define USBO ((USB_Type *)USBO_BASE)

#define USBO_BASE_PTR (USBO)

/** Array initializer of USB peripheral base addresses */

#define USB_BASE_ADDRS { USBO_BASE }

/** Array initializer of USB peripheral base pointers */

#define USB_BASE_PTRS { USBO }

/*
-- USB - Register accessor macros
*/
/*!
* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros
* @{
*/

```

```

/* USB - Register instance definitions */

/* USBO */

#define USBO_PERID USB_PERID_REG(USBO)
#define USBO_IDCOMP USB_IDCOMP_REG(USBO)
#define USBO_REV USB_REV_REG(USBO)
#define USBO_ADDINFO USB_ADDINFO_REG(USBO)
#define USBO_OTGISTAT USB_OTGISTAT_REG(USBO)
#define USBO_OTGICR USB_OTGICR_REG(USBO)
#define USBO_OTGSTAT USB_OTGSTAT_REG(USBO)
#define USBO_OTGCTL USB_OTGCTL_REG(USBO)
#define USBO_ISTAT USB_ISTAT_REG(USBO)
#define USBO_INTEN USB_INTEN_REG(USBO)

```

#define USBO_ERRSTAT	USB_ERRSTAT_REG(USBO)
#define USBO_ERREN	USB_ERREN_REG(USBO)
#define USBO_STAT	USB_STAT_REG(USBO)
#define USBO_CTL	USB_CTL_REG(USBO)
#define USBO_ADDR	USB_ADDR_REG(USBO)
#define USBO_BDTPAGE1	USB_BDTPAGE1_REG(USBO)
#define USBO_FRMNUML	USB_FRMNUML_REG(USBO)
#define USBO_FRMNUMH	USB_FRMNUMH_REG(USBO)
#define USBO_TOKEN	USB_TOKEN_REG(USBO)
#define USBO_SOFTHLD	USB_SOFTHLD_REG(USBO)
#define USBO_BDTPAGE2	USB_BDTPAGE2_REG(USBO)
#define USBO_BDTPAGE3	USB_BDTPAGE3_REG(USBO)
#define USBO_ENDPT0	USB_ENDPT_REG(USBO,0)
#define USBO_ENDPT1	USB_ENDPT_REG(USBO,1)
#define USBO_ENDPT2	USB_ENDPT_REG(USBO,2)
#define USBO_ENDPT3	USB_ENDPT_REG(USBO,3)
#define USBO_ENDPT4	USB_ENDPT_REG(USBO,4)
#define USBO_ENDPT5	USB_ENDPT_REG(USBO,5)
#define USBO_ENDPT6	USB_ENDPT_REG(USBO,6)
#define USBO_ENDPT7	USB_ENDPT_REG(USBO,7)
#define USBO_ENDPT8	USB_ENDPT_REG(USBO,8)
#define USBO_ENDPT9	USB_ENDPT_REG(USBO,9)
#define USBO_ENDPT10	USB_ENDPT_REG(USBO,10)
#define USBO_ENDPT11	USB_ENDPT_REG(USBO,11)
#define USBO_ENDPT12	USB_ENDPT_REG(USBO,12)
#define USBO_ENDPT13	USB_ENDPT_REG(USBO,13)
#define USBO_ENDPT14	USB_ENDPT_REG(USBO,14)
#define USBO_ENDPT15	USB_ENDPT_REG(USBO,15)
#define USBO_USBCTRL	USB_USBCTRL_REG(USBO)

```
#define USBO_OBSERVE           USB_OBSERVE_REG(USBO)
#define USBO_CONTROL            USB_CONTROL_REG(USBO)
#define USBO_USBTRC0            USB_USBTRC0_REG(USBO)
#define USBO_USBFRMADJUST       USB_USBFRMADJUST_REG(USBO)
```

```
/* USB - Register array accessors */
```

```
#define USBO_ENDPT(index)      USB_ENDPT_REG(USBO,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Peripheral_Access_Layer */
```

```
/*
```

```
** End of section using anonymous unions
```

```
*/
```

```
#if defined(__ARMCC_VERSION)
    #pragma pop
#elif defined(__CWCC__)
    #pragma pop
#elif defined(__GNUC__)
    /* leave anonymous unions enabled */
#elif defined(__IAR_SYSTEMS_ICC__)
```

```

#pragma language=default

#else
#error Not supported compiler type

#endif

/*!
 * @}
 */ /* end of group Peripheral_access_layer */

/*
-----  

-- Backward Compatibility  

----- */
/*!

 * @addtogroup Backward_Compatibility_Symbols Backward Compatibility
 * @{
 *
#define DMA_REQC_ARR_DMAC_MASK      This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC_SHIFT     This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC(x)        This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_MASK      This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_SHIFT     This_symbol_has_been_deprecated
#define DMA_REQC0                   This_symbol_has_been_deprecated
#define DMA_REQC1                   This_symbol_has_been_deprecated
#define DMA_REQC2                   This_symbol_has_been_deprecated
#define DMA_REQC3                   This_symbol_has_been_deprecated
#define MCG_S_LOLS_MASK            MCG_S_LOLS0_MASK

```

#define MCG_S_LOLS_SHIFT	MCG_S_LOLS0_SHIFT
#define SIM_FCFG2_MAXADDR_MASK	SIM_FCFG2_MAXADDR0_MASK
#define SIM_FCFG2_MAXADDR_SHIFT	SIM_FCFG2_MAXADDR0_SHIFT
#define SIM_FCFG2_MAXADDR	SIM_FCFG2_MAXADDR0
#define SPI_C2_SPLPIE_MASK	This_symbol_has_been_deprecated
#define SPI_C2_SPLPIE_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_SHIFT	This_symbol_has_been_deprecated
#define UARTLP_Type	UART0_Type
#define UARTLP_BDH_REG	UART0_BDH_REG
#define UARTLP_BDL_REG	UART0_BDL_REG
#define UARTLP_C1_REG	UART0_C1_REG
#define UARTLP_C2_REG	UART0_C2_REG
#define UARTLP_S1_REG	UART0_S1_REG
#define UARTLP_S2_REG	UART0_S2_REG
#define UARTLP_C3_REG	UART0_C3_REG
#define UARTLP_D_REG	UART0_D_REG
#define UARTLP_MA1_REG	UART0_MA1_REG
#define UARTLP_MA2_REG	UART0_MA2_REG
#define UARTLP_C4_REG	UART0_C4_REG
#define UARTLP_C5_REG	UART0_C5_REG
#define UARTLP_BDH_SBR_MASK	UART0_BDH_SBR_MASK
#define UARTLP_BDH_SBR_SHIFT	UART0_BDH_SBR_SHIFT
#define UARTLP_BDH_SBR(x)	UART0_BDH_SBR(x)
#define UARTLP_BDH_SBNS_MASK	UART0_BDH_SBNS_MASK

#define UARLTP_BDH_SBNS_SHIFT	UART0_BDH_SBNS_SHIFT
#define UARLTP_BDH_RXEDGIE_MASK	UART0_BDH_RXEDGIE_MASK
#define UARLTP_BDH_RXEDGIE_SHIFT	UART0_BDH_RXEDGIE_SHIFT
#define UARLTP_BDH_LBKDIE_MASK	UART0_BDH_LBKDIE_MASK
#define UARLTP_BDH_LBKDIE_SHIFT	UART0_BDH_LBKDIE_SHIFT
#define UARLTP_BDL_SBR_MASK	UART0_BDL_SBR_MASK
#define UARLTP_BDL_SBR_SHIFT	UART0_BDL_SBR_SHIFT
#define UARLTP_BDL_SBR(x)	UART0_BDL_SBR(x)
#define UARLTP_C1_PT_MASK	UART0_C1_PT_MASK
#define UARLTP_C1_PT_SHIFT	UART0_C1_PT_SHIFT
#define UARLTP_C1_PE_MASK	UART0_C1_PE_MASK
#define UARLTP_C1_PE_SHIFT	UART0_C1_PE_SHIFT
#define UARLTP_C1_ILT_MASK	UART0_C1_ILT_MASK
#define UARLTP_C1_ILT_SHIFT	UART0_C1_ILT_SHIFT
#define UARLTP_C1_WAKE_MASK	UART0_C1_WAKE_MASK
#define UARLTP_C1_WAKE_SHIFT	UART0_C1_WAKE_SHIFT
#define UARLTP_C1_M_MASK	UART0_C1_M_MASK
#define UARLTP_C1_M_SHIFT	UART0_C1_M_SHIFT
#define UARLTP_C1_RSRC_MASK	UART0_C1_RSRC_MASK
#define UARLTP_C1_RSRC_SHIFT	UART0_C1_RSRC_SHIFT
#define UARLTP_C1_DOZEEN_MASK	UART0_C1_DOZEEN_MASK
#define UARLTP_C1_DOZEEN_SHIFT	UART0_C1_DOZEEN_SHIFT
#define UARLTP_C1_LOOPS_MASK	UART0_C1_LOOPS_MASK
#define UARLTP_C1_LOOPS_SHIFT	UART0_C1_LOOPS_SHIFT
#define UARLTP_C2_SBK_MASK	UART0_C2_SBK_MASK
#define UARLTP_C2_SBK_SHIFT	UART0_C2_SBK_SHIFT
#define UARLTP_C2_RWU_MASK	UART0_C2_RWU_MASK
#define UARLTP_C2_RWU_SHIFT	UART0_C2_RWU_SHIFT
#define UARLTP_C2_RE_MASK	UART0_C2_RE_MASK

#define UARTLP_C2_RE_SHIFT	UART0_C2_RE_SHIFT
#define UARTLP_C2_TE_MASK	UART0_C2_TE_MASK
#define UARTLP_C2_TE_SHIFT	UART0_C2_TE_SHIFT
#define UARTLP_C2_ILIE_MASK	UART0_C2_ILIE_MASK
#define UARTLP_C2_ILIE_SHIFT	UART0_C2_ILIE_SHIFT
#define UARTLP_C2_RIE_MASK	UART0_C2_RIE_MASK
#define UARTLP_C2_RIE_SHIFT	UART0_C2_RIE_SHIFT
#define UARTLP_C2_TCIE_MASK	UART0_C2_TCIE_MASK
#define UARTLP_C2_TCIE_SHIFT	UART0_C2_TCIE_SHIFT
#define UARTLP_C2_TIE_MASK	UART0_C2_TIE_MASK
#define UARTLP_C2_TIE_SHIFT	UART0_C2_TIE_SHIFT
#define UARTLP_S1_PF_MASK	UART0_S1_PF_MASK
#define UARTLP_S1_PF_SHIFT	UART0_S1_PF_SHIFT
#define UARTLP_S1_FE_MASK	UART0_S1_FE_MASK
#define UARTLP_S1_FE_SHIFT	UART0_S1_FE_SHIFT
#define UARTLP_S1_NF_MASK	UART0_S1_NF_MASK
#define UARTLP_S1_NF_SHIFT	UART0_S1_NF_SHIFT
#define UARTLP_S1_OR_MASK	UART0_S1_OR_MASK
#define UARTLP_S1_OR_SHIFT	UART0_S1_OR_SHIFT
#define UARTLP_S1_IDLE_MASK	UART0_S1_IDLE_MASK
#define UARTLP_S1_IDLE_SHIFT	UART0_S1_IDLE_SHIFT
#define UARTLP_S1_RDRF_MASK	UART0_S1_RDRF_MASK
#define UARTLP_S1_RDRF_SHIFT	UART0_S1_RDRF_SHIFT
#define UARTLP_S1_TC_MASK	UART0_S1_TC_MASK
#define UARTLP_S1_TC_SHIFT	UART0_S1_TC_SHIFT
#define UARTLP_S1_TDRE_MASK	UART0_S1_TDRE_MASK
#define UARTLP_S1_TDRE_SHIFT	UART0_S1_TDRE_SHIFT
#define UARTLP_S2_RAF_MASK	UART0_S2_RAF_MASK
#define UARTLP_S2_RAF_SHIFT	UART0_S2_RAF_SHIFT

#define UARLTP_S2_LBKDE_MASK	UART0_S2_LBKDE_MASK
#define UARLTP_S2_LBKDE_SHIFT	UART0_S2_LBKDE_SHIFT
#define UARLTP_S2_BRK13_MASK	UART0_S2_BRK13_MASK
#define UARLTP_S2_BRK13_SHIFT	UART0_S2_BRK13_SHIFT
#define UARLTP_S2_RWUID_MASK	UART0_S2_RWUID_MASK
#define UARLTP_S2_RWUID_SHIFT	UART0_S2_RWUID_SHIFT
#define UARLTP_S2_RXINV_MASK	UART0_S2_RXINV_MASK
#define UARLTP_S2_RXINV_SHIFT	UART0_S2_RXINV_SHIFT
#define UARLTP_S2_MSBF_MASK	UART0_S2_MSBF_MASK
#define UARLTP_S2_MSBF_SHIFT	UART0_S2_MSBF_SHIFT
#define UARLTP_S2_RXEDGIF_MASK	UART0_S2_RXEDGIF_MASK
#define UARLTP_S2_RXEDGIF_SHIFT	UART0_S2_RXEDGIF_SHIFT
#define UARLTP_S2_LBKDIF_MASK	UART0_S2_LBKDIF_MASK
#define UARLTP_S2_LBKDIF_SHIFT	UART0_S2_LBKDIF_SHIFT
#define UARLTP_C3_PEIE_MASK	UART0_C3_PEIE_MASK
#define UARLTP_C3_PEIE_SHIFT	UART0_C3_PEIE_SHIFT
#define UARLTP_C3_FEIE_MASK	UART0_C3_FEIE_MASK
#define UARLTP_C3_FEIE_SHIFT	UART0_C3_FEIE_SHIFT
#define UARLTP_C3_NEIE_MASK	UART0_C3_NEIE_MASK
#define UARLTP_C3_NEIE_SHIFT	UART0_C3_NEIE_SHIFT
#define UARLTP_C3_ORIE_MASK	UART0_C3_ORIE_MASK
#define UARLTP_C3_ORIE_SHIFT	UART0_C3_ORIE_SHIFT
#define UARLTP_C3_TXINV_MASK	UART0_C3_TXINV_MASK
#define UARLTP_C3_TXINV_SHIFT	UART0_C3_TXINV_SHIFT
#define UARLTP_C3_TXDIR_MASK	UART0_C3_TXDIR_MASK
#define UARLTP_C3_TXDIR_SHIFT	UART0_C3_TXDIR_SHIFT
#define UARLTP_C3_R9T8_MASK	UART0_C3_R9T8_MASK
#define UARLTP_C3_R9T8_SHIFT	UART0_C3_R9T8_SHIFT
#define UARLTP_C3_R8T9_MASK	UART0_C3_R8T9_MASK

#define UARTLP_C3_R8T9_SHIFT	UART0_C3_R8T9_SHIFT
#define UARTLP_DROTO_MASK	UART0_DROTO_MASK
#define UARTLP_DROTO_SHIFT	UART0_DROTO_SHIFT
#define UARTLP_DR1T1_MASK	UART0_DR1T1_MASK
#define UARTLP_DR1T1_SHIFT	UART0_DR1T1_SHIFT
#define UARTLP_DR2T2_MASK	UART0_DR2T2_MASK
#define UARTLP_DR2T2_SHIFT	UART0_DR2T2_SHIFT
#define UARTLP_DR3T3_MASK	UART0_DR3T3_MASK
#define UARTLP_DR3T3_SHIFT	UART0_DR3T3_SHIFT
#define UARTLP_DR4T4_MASK	UART0_DR4T4_MASK
#define UARTLP_DR4T4_SHIFT	UART0_DR4T4_SHIFT
#define UARTLP_DR5T5_MASK	UART0_DR5T5_MASK
#define UARTLP_DR5T5_SHIFT	UART0_DR5T5_SHIFT
#define UARTLP_DR6T6_MASK	UART0_DR6T6_MASK
#define UARTLP_DR6T6_SHIFT	UART0_DR6T6_SHIFT
#define UARTLP_DR7T7_MASK	UART0_DR7T7_MASK
#define UARTLP_DR7T7_SHIFT	UART0_DR7T7_SHIFT
#define UARTLP_MA1_MA_MASK	UART0_MA1_MA_MASK
#define UARTLP_MA1_MA_SHIFT	UART0_MA1_MA_SHIFT
#define UARTLP_MA1_MA(x)	UART0_MA1_MA(x)
#define UARTLP_MA2_MA_MASK	UART0_MA2_MA_MASK
#define UARTLP_MA2_MA_SHIFT	UART0_MA2_MA_SHIFT
#define UARTLP_MA2_MA(x)	UART0_MA2_MA(x)
#define UARTLP_C4_OSR_MASK	UART0_C4_OSR_MASK
#define UARTLP_C4_OSR_SHIFT	UART0_C4_OSR_SHIFT
#define UARTLP_C4_OSR(x)	UART0_C4_OSR(x)
#define UARTLP_C4_M10_MASK	UART0_C4_M10_MASK
#define UARTLP_C4_M10_SHIFT	UART0_C4_M10_SHIFT
#define UARTLP_C4_MAEN2_MASK	UART0_C4_MAEN2_MASK

```
#define UARTLP_C4_MAEN2_SHIFT           UART0_C4_MAEN2_SHIFT
#define UARTLP_C4_MAEN1_MASK              UART0_C4_MAEN1_MASK
#define UARTLP_C4_MAEN1_SHIFT             UART0_C4_MAEN1_SHIFT
#define UARTLP_C5_RESETDISC_MASK          UART0_C5_RESETDISC_MASK
#define UARTLP_C5_RESETDISC_SHIFT         UART0_C5_RESETDISC_SHIFT
#define UARTLP_C5_BOTEDGE_MASK            UART0_C5_BOTEDGE_MASK
#define UARTLP_C5_BOTEDGE_SHIFT           UART0_C5_BOTEDGE_SHIFT
#define UARTLP_C5_RDMAE_MASK              UART0_C5_RDMAE_MASK
#define UARTLP_C5_RDMAE_SHIFT             UART0_C5_RDMAE_SHIFT
#define UARTLP_C5_TDMAE_MASK              UART0_C5_TDMAE_MASK
#define UARTLP_C5_TDMAE_SHIFT             UART0_C5_TDMAE_SHIFT

#define UARTLP_BASES                     UARTLP_BASES
#define NV_FOPT_EZPORT_DIS_MASK          This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT         This_symbol_has_been_deprecated
#define ADC_BASES                        ADC_BASE_PTRS
#define CMP_BASES                        CMP_BASE_PTRS
#define DAC_BASES                        DAC_BASE_PTRS
#define DMA_BASES                        DMA_BASE_PTRS
#define DMAMUX_BASES                    DMAMUX_BASE_PTRS
#define FPTA_BASE_PTR                   FGPIOA_BASE_PTR
#define FPTA_BASE                        FGPIOA_BASE
#define FPTA                           FGPIOA
#define FPTB_BASE_PTR                   FGPIOB_BASE_PTR
#define FPTB_BASE                        FGPIOB_BASE
#define FPTB                           FGPIOB
#define FPTC_BASE_PTR                   FGPIOC_BASE_PTR
#define FPTC_BASE                        FGPIOC_BASE
#define FPTC                           FGPIOC
#define FPTD_BASE_PTR                   FGPIOD_BASE_PTR
```

#define FPTD_BASE	FGPIOD_BASE
#define FPTD	FGPIOD
#define FPTE_BASE_PTR	FGPIOE_BASE_PTR
#define FPTE_BASE	FGPIOE_BASE
#define FPTE	FGPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define FTFA_BASES	FTFA_BASE_PTRS
#define PTA_BASE_PTR	GPIOA_BASE_PTR
#define PTA_BASE	GPIOA_BASE
#define PTA	GPIOA
#define PTB_BASE_PTR	GPIOB_BASE_PTR
#define PTB_BASE	GPIOB_BASE
#define PTB	GPIOB
#define PTC_BASE_PTR	GPIOC_BASE_PTR
#define PTC_BASE	GPIOC_BASE
#define PTC	GPIOC
#define PTD_BASE_PTR	GPIOD_BASE_PTR
#define PTD_BASE	GPIOD_BASE
#define PTD	GPIOD
#define PTE_BASE_PTR	GPIOE_BASE_PTR
#define PTE_BASE	GPIOE_BASE
#define PTE	GPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define I2C_BASES	I2C_BASE_PTRS
#define LLWU_BASES	LLWU_BASE_PTRS
#define LPTMR_BASES	LPTMR_BASE_PTRS
#define MCG_BASES	MCG_BASE_PTRS
#define MCM_BASES	MCM_BASE_PTRS
#define MTB_BASES	MTB_BASE_PTRS

```
#define MTBDWT_BASES           MTBDWT_BASE_PTRS
#define NV_BASES                 NV_BASES
#define OSC_BASES                OSC_BASE_PTRS
#define PIT_BASES                PIT_BASE_PTRS
#define PMC_BASES                PMC_BASE_PTRS
#define PORT_BASES               PORT_BASE_PTRS
#define RCM_BASES                RCM_BASE_PTRS
#define ROM_BASES                ROM_BASE_PTRS
#define RTC_BASES                RTC_BASE_PTRS
#define SIM_BASES                SIM_BASE_PTRS
#define SMC_BASES                SMC_BASE_PTRS
#define SPI_BASES                SPI_BASE_PTRS
#define TPM_BASES                TPM_BASE_PTRS
#define TSI_BASES                TSI_BASE_PTRS
#define UART_BASES               UART_BASE_PTRS
#define UART0_BASES              UART0_BASE_PTRS
#define USB_BASES                USB_BASE_PTRS
#define LPTimer_IRQn             LPTMR0_IRQn
#define LPTimer_IRQHandler        LPTMR0_IRQHandler
#define LLW_IRQn                 LLWU_IRQn
#define LLW_IRQHandler           LLWU_IRQHandler
```

```
/*
 * @}
 */
/* /* end of group Backward_Compatibility_Symbols */
```

```
#else /* #if !defined(MKL25Z4_H_) */
```

```
/* There is already included the same memory map. Check if it is compatible (has the same major
version) */

#if (MCU_MEM_MAP_VERSION != 0x0200u)

#if (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING))

#warning There are included two not compatible versions of memory maps. Please check possible
differences.

#endif /* (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING)) */

#endif /* (MCU_MEM_MAP_VERSION != 0x0200u) */

#endif /* #if !defined(MKL25Z4_H_) */

/* MKL25Z4.h, eof. */

/*
** #####
** Processors:      MKL25Z128FM4
**                  MKL25Z128FT4
**                  MKL25Z128LH4
**                  MKL25Z128VLK4
**
** Compilers:       Keil ARM C/C++ Compiler
**                  Freescale C/C++ for Embedded ARM
**                  GNU C Compiler
**                  GNU C Compiler - CodeSourcery Sourcery G++
**                  IAR ANSI C/C++ Compiler for ARM
**
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012
** Version:         rev. 2.5, 2015-02-19
** Build:          b150220
**
```

** Abstract:

** Provides a system configuration function and a global variable that

** contains the system frequency. It configures the device and initializes

** the oscillator (PLL) that is part of the microcontroller device.

**

** Copyright (c) 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com

** mail: support@freescale.com

**

** Revisions:

** - rev. 1.0 (2012-06-13)

** Initial version.

** - rev. 1.1 (2012-06-21)

** Update according to reference manual rev. 1.

** - rev. 1.2 (2012-08-01)

** Device type UARTLP changed to UART0.

** - rev. 1.3 (2012-10-04)

** Update according to reference manual rev. 3.

** - rev. 1.4 (2012-11-22)

** MCG module - bit LOLS in MCG_S register renamed to LOLS0.

** NV registers - bit EZPORT_DIS in NV_FOPT register removed.

** - rev. 1.5 (2013-04-05)

** Changed start of doxygen comment.

** - rev. 2.0 (2013-10-29)

** Register accessor macros added to the memory map.

** Symbols for Processor Expert memory map compatibility added to the memory map.

** Startup file for gcc has been updated according to CMSIS 3.2.

** System initialization updated.

** - rev. 2.1 (2014-07-16)

** Module access macro module_BASES replaced by module_BASE_PTRS.

** System initialization and startup updated.

```
** - rev. 2.2 (2014-08-22)
** System initialization updated - default clock config changed.

** - rev. 2.3 (2014-08-28)
** Update of startup files - possibility to override DefaultISR added.

** - rev. 2.4 (2014-10-14)
** Interrupt INT_LPTimer renamed to INT_LPTMR0.

** - rev. 2.5 (2015-02-19)
** Renamed interrupt vector LLW to LLWU.

**
** #####
*/

```

```
/*!
* @file MKL25Z4
* @version 2.5
* @date 2015-02-19
* @brief Device specific configuration file for MKL25Z4 (header file)
*
* Provides a system configuration function and a global variable that contains
* the system frequency. It configures the device and initializes the oscillator
* (PLL) that is part of the microcontroller device.
*/

```

```
#ifndef SYSTEM_MKL25Z4_H_
#define SYSTEM_MKL25Z4_H_          /**< Symbol preventing repeated inclusion */
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
#include <stdint.h>

#ifndef DISABLE_WDOG
#define DISABLE_WDOG      1
#endif

/* MCG mode constants */

#define MCG_MODE_FEI      0U
#define MCG_MODE_FBI      1U
#define MCG_MODE_BLPI     2U
#define MCG_MODE_FEE      3U
#define MCG_MODE_FBE      4U
#define MCG_MODE_BLPE     5U
#define MCG_MODE_PBE      6U
#define MCG_MODE_PEE      7U

/* Predefined clock setups

0 ... Default part configuration
    Multipurpose Clock Generator (MCG) in FEI mode.
    Reference clock source for MCG module: Slow internal reference clock
    Core clock = 20.97152MHz
    Bus clock = 20.97152MHz

1 ... Maximum achievable clock frequency configuration
```

Multipurpose Clock Generator (MCG) in PEE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 48MHz

Bus clock = 24MHz

2 ... Chip internally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPI mode.

Reference clock source for MCG module: Fast internal reference clock

Core clock = 4MHz

Bus clock = 0.8MHz

3 ... Chip externally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 4MHz

Bus clock = 1MHz

4 ... USB clock setup

Multipurpose Clock Generator (MCG) in PEE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 48MHz

Bus clock = 24MHz

*/

/* Define clock source values */

```
#define CPU_XTAL_CLK_HZ      8000000u      /* Value of the external crystal or oscillator clock  
frequency in Hz */
```

```
#define CPU_INT_SLOW_CLK_HZ   32768u       /* Value of the slow internal oscillator clock  
frequency in Hz */
```

```
#define CPU_INT_FAST_CLK_HZ   4000000u     /* Value of the fast internal oscillator clock  
frequency in Hz */
```

```

/* RTC oscillator setting */

/* Low power mode enable */

/* SMC_PMPROT: AVLP=1,ALLS=1,AVLLS=1 */

#define SYSTEM_SMC_PMPROT_VALUE      0x2AU      /* SMC_PMPROT */

/* Internal reference clock trim */

/* #undef SLOW_TRIM_ADDRESS */          /* Slow oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef SLOW_FINE_TRIM_ADDRESS */     /* Slow oscillator not trimmed. Commented out
for MISRA compliance. */

/* #undef FAST_TRIM_ADDRESS */         /* Fast oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef FAST_FINE_TRIM_ADDRESS */    /* Fast oscillator not trimmed. Commented out
for MISRA compliance. */

#endif CLOCK_SETUP

#if (CLOCK_SETUP == 0)

#define DEFAULT_SYSTEM_CLOCK      20971520u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_FEI /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE      0x06U      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSS0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE      0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCSS0=0 */

#define SYSTEM_MCG_SC_VALUE      0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE      0x00U      /* MCG_C5 */

```

```

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE      0x00U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=0 */
#define SYSTEM_SIM_CLKDIV1_VALUE  0x00U      /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U /* SIM_SOPT1 */

/* SIM_SOPT2: UARTOSRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE    0x01000000U /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 1)

#define DEFAULT_SYSTEM_CLOCK      48000000u /* Default System clock value */

#define MCG_MODE                  MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE       0x1AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HG00=0,EREFS0=1,LP=0,IRCS=0 */
#define SYSTEM_MCG_C2_VALUE       0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRs=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE       0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCs0=0 */
#define SYSTEM_MCG_SC_VALUE       0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=1 */
#define SYSTEM_MCG_C5_VALUE       0x01U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE       0x40U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

```

```

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE    0x00U           /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */

#define SYSTEM_SIM_CLKDIV1_VALUE   0x10010000U     /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U     /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U     /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 2)

#define DEFAULT_SYSTEM_CLOCK      4000000u        /* Default System clock value */

#define MCG_MODE                  MCG_MODE_BLPI /* Clock generator mode */

/* MCG_C1: CLKS=1,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE       0x46U           /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSO=1,LP=1,IRCS=1 */

#define SYSTEM_MCG_C2_VALUE       0x27U           /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE       0x00U           /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSrv=0,FCRDIV=0,LOCs0=0 */

#define SYSTEM_MCG_SC_VALUE       0x00U           /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE       0x00U           /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE       0x00U           /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE      0x80U           /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U           /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=4 */

#define SYSTEM_SIM_CLKDIV1_VALUE  0x00040000U     /* SIM_CLKDIV1 */

```

```

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x02000000U    /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 3)

#define DEFAULT_SYSTEM_CLOCK     4000000u      /* Default System clock value */

#define MCG_MODE                 MCG_MODE_BLPE /* Clock generator mode */

/* MCG_C1: CLKS=2,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE     0x9AU        /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFS0=1,LP=1,IRCS=1 */

#define SYSTEM_MCG_C2_VALUE     0x27U        /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRs=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE     0x00U        /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=1,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE     0x02U        /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE     0x00U        /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE     0x00U        /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE    0x80U        /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE  0x00U        /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=3 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x10030000U   /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x02000000U    /* SIM_SOPT2 */

```

```

#endif (CLOCK_SETUP == 4)

#define DEFAULT_SYSTEM_CLOCK      48000000u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE      0x1AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFST0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE      0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DR=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE      0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=1 */

#define SYSTEM_MCG_C5_VALUE      0x01U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE      0x40U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */

#define SYSTEM_SIM_CLKDIV1_VALUE  0x10010000U    /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U    /* SIM_SOPT2 */

```

```
#endif

/** 
 * @brief System clock frequency (core clock)
 *
 * The system clock frequency supplied to the SysTick timer and the processor
 * core clock. This variable can be used by the user application to setup the
 * SysTick timer or configure other parameters. It may also be used by debugger to
 * query the frequency of the debug timer or configure the trace clock speed
 * SystemCoreClock is initialized with a correct predefined value.
 */
extern uint32_t SystemCoreClock;

/** 
 * @brief Setup the microcontroller system.
 *
 * Typically this function configures the oscillator (PLL) that is part of the
 * microcontroller device. For systems with variable clock speed it also updates
 * the variable SystemCoreClock. SystemInit is called from startup_device file.
 */
void SystemInit (void);

/** 
 * @brief Updates the SystemCoreClock variable.
 *
 * It must be called whenever the core clock is changed during program
 * execution. SystemCoreClockUpdate() evaluates the clock register settings and calculates
 * the current core clock.
 */

```

```
void SystemCoreClockUpdate (void);

#ifndef __cplusplus
}

#endif /* #if !defined(SYSTEM_MKL25Z4_H_) */

/*
 * circbuf.c
 *
 * Created on: Oct 28, 2017
 * Author: harsi
 */

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include "circbuf.h"

CB_status CB_init(CB_t **cbuf,uint8_t length)
{
    CB_t *temp = *cbuf;
    temp->buffer = NULL;
    temp->buffer = malloc(sizeof(uint8_t)*length);
    temp->length = length;
    temp->head = temp->buffer;
    temp->tail = temp->buffer;
    temp->count = 0;
    temp->peek = 0;
    *cbuf=temp;

    return No_error;
```

```
}
```

```
CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val)
{ if(!cbuf->tail)
{
    return Null_pointer;
}
if(cbuf->count == cbuf->length)
{
    return Full;
}
*(cbuf->tail) = val;
if(cbuf->tail==cbuf->buffer+(cbuf->length-1))
{
    cbuf->tail = cbuf->buffer;
}
else
{
    cbuf->tail++;
}
cbuf->count++;

return No_error;
}

CB_status CB_buffer_remove_item(CB_t *cbuf /*uint8_t *value*/)
{
if(cbuf->count==0)
{
    return Empty;
}
```

```

//      *value = *(cbuf->tail);

*(cbuf->head)=100;

if(cbuf->head == cbuf->buffer+cbuf->length-sizeof(uint8_t))

{

    cbuf->head = cbuf->buffer;

}

else

{

    cbuf->head++;

}

cbuf->count = cbuf->count-1;

cbuf->status=No_error;

return No_error;

}

```

```

CB_status CB_is_full(CB_t *cbuf)

{

if(cbuf == NULL)

{

    return Null_pointer;

}

if(cbuf->count == cbuf->length)

{

    return Full;

}

else

    return Available;

}

```

```
CB_status CB_is_empty(CB_t *cbuf)
{
    if(cbuf == NULL)
    {
        return Null_pointer;
    }
    if(cbuf->count == 0)
    {
        return Empty;
    }
    else
        return Available;
}
```

```
CB_status CB_peek(CB_t *cbuf, uint8_t position)
{
    if(cbuf == NULL)
    {
        return Null_pointer;
    }
    else if(position > cbuf->length)
    {
        return Wrong_memory_access;
    }
    else if(position > cbuf->count)
    {
        return Empty;
    }
    else
```

```

{
    volatile uint8_t *pos = cbuf->tail;

    while(position)
    {
        if(pos == cbuf->buffer+cbuf->length-sizeof(uint8_t))
        {
            pos = cbuf->buffer;
        }
        else
        {
            pos = pos+1;
        }
        position--;
    }

    cbuf->peek = *pos;
}

return No_error;
}

```

```

CB_status CB_destroy(CB_t *cbuf)
{
    free((uint8_t *)cbuf->buffer);
    free(cbuf);

    return No_error;
}

```

```
*****
```

```
* * Authors: Harsimransingh Bindra, Smitesh Modak
```

```
* * Date Edited: 10/29/2017
*
* * File: uart.c
*
* * Description: Header file for uart communication
* * -UART_config
* * -UART_send
* * -UART_send_n
* * -UART_recieve
* * -UART_receive_n
* * -UART0_IRQHandler
*
* *
* *
* *****/
```

```
#include <stdio.h>
#include <stdlib.h>
#include "uart.h"
#include <MKL25Z4.H>
#include "circbuf.h"

#define MCGFLLCLK 0x04000000
#define OSR_16 0x0F
#define BDH_BAUD_VALUE 0x00
#define BDL_BAUD_57600 0x17
#define buffsize 100

CB_t *transbuf;
```

```

CB_t *recbuf;

uint8_t recflag = 0;

CB_t *create_buffer(CB_t *cbuf)
{
    cbuf = malloc(sizeof(CB_t));
    CB_init(&cbuf,100);
    return cbuf;
}

int UART_configure(void)
{
    /*Provide clock to UART0*/
    SIM_SCGC4 |= SIM_SCGC4_UART0_MASK;

    /*Selecting MCGFLL clock for UART0*/
    SIM_SOPT2 |= SIM_SOPT2_UART0SRC_MASK & MCGFLLCLK;

    /*Turn Off UART0 to make changes and then enable it */
    UART0_C2 = 0x00;

    /*Adjusting values of BDH and BDL to set baud rate as 57600 */
    UART0_BDH = BDH_BAUD_VALUE;
    UART0_BDL = BDL_BAUD_57600;

    UART0_C4 = UART0_C4_OSRSR_MASK & OSRSR_16;

    UART0_C1 = 0x00;
}

```

```

SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

PORTA_PCR2 = 0x0200;
PORTA_PCR1 = 0x0200;

NVIC->ISER[0] |= 0x00001000; /* enable INT12 (bit 12 of ISER[0]) */
NVIC_ClearPendingIRQ(UART0_IRQn);
NVIC_EnableIRQ(UART0_IRQn);
NVIC_SetPriority(UART0_IRQn, 2);
UART0_C2 = UART0_C2_RX_MASK | UART0_C2_TE_MASK | UART0_C2_RIE_MASK;
}

CB_status add_to_buffer(uint8_t *str_ptr)
{
    while(*str_ptr != '\0') //checks for the null character
    {
        CB_buffer_add_item(transbuf,*str_ptr); //adds the data to the transmit buffer
        str_ptr++; // increments the pointer
    }
    return No_error;
}

void UART_send(uint8_t *data)
{
    while(!(UART0->S1 & 0xC0))
    {

```

```

        }

/* to wait until
the transmitting buffer is empty */

UART0_D = *data;                                //write data in the UART0_D
register to transmit

}

void UART_send_n(uint8_t *data, uint8_t length)
{
    for (int i=0;i<=length-1;i++)                //Loop to execute transmitting of data based on
the length.

    {
        while(!(UART0->S1 & 0xC0))

        {

            }                                         /* wait for
transmit buffer empty */

        UART0_D = *data;                            //write data in the UART0_D
register to transmit

        data++;

    }

}

void UART_receive(uint8_t *data)
{
    uint8_t data=0;

    while(!(UART0->S1 & 0x20))

    {

    } /* wait for receive buffer full */

    *data = UART0_D ;                           // read the char received

    return data;

}

```

```

void UART_receive_n(uint8_t *data,uint8_t length)      //Loop to execute receiving of data based on
the length.

{
    for(int i=0;i<=length-1;i++)
    {
        while(!(UART0->S1 & 0x20))

        {
            /* wait for
receive buffer full */

            *data = UART0_D ;           // read the char received
            data++;
        }
    }
}

```

```

void UART0_IRQHandler(void)
{
    __disable_irq();           //disable interrupt once received

    if(((UART0_C2 & UART0_C2_TIE_MASK)!=0) && (transbuf->count != 0)){ //check if
transmit interrupt is enabled and buffer count is not equal to 0

        UART_send(transbuf->head);
        //transmit data

        CB_buffer_remove_item(transbuf/*, &UART0_D*/);
        //remove transmitted data

        if(transbuf->count == 0)

    {

        UART0_C2 &= ~UART0_C2_TIE_MASK;
        //disable transmit interrupt

    }

    else

        UART0_C2 |= UART0_C2_TIE_MASK;
        //keep transmit interrupt enabled
}

```

```

    }

    else if(((UART0_C2 & UART0_C2_RIE_MASK)!=0) && ((UART0_S1 &
UART0_S1_RDRF_MASK) !=0)){ //check if receive interrupt is enabled

        uint8_t val = UART_receive();
        //read received data

        CB_buffer_add_item(recbuf,val);
        //add data to receive buffer

        //UART0_C2 &= ~UART_C2_RIE_MASK;

        recflag = 1;
        //set receive flag

    }

    __enable_irq();
    //enable interrupt once process for previous interrupt is done
}

```

/*

* Copyright (c) 2015, Freescale Semiconductor, Inc.

* All rights reserved.

*

* Redistribution and use in source and binary forms, with or without modification,

* are permitted provided that the following conditions are met:

*

* o Redistributions of source code must retain the above copyright notice, this list

* of conditions and the following disclaimer.

*

* o 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.
*
* o Neither the name of Freescale Semiconductor, Inc. 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.
*/

```
#include <stdlib.h>
#include "MKL25Z4.h"
#include "uart.h"
#include "circbuf.h"
#include "project2.h"
```

```
extern CB_t *transbuf;
extern CB_t *recbuf;
extern uint8_t recflag;
int main(void)
```

```

{
uint8_t i = 0;
transbuf = create_buffer(transbuf);
recbuf = create_buffer(recbuf);
uint8_t val = 0;
UART_configure();
uint8_t str[] = "harsimrarsingh Bindra";
add_to_buffer(str);
UART0_C2 |= UART0_C2_TIE_MASK | UART0_C2_RIE_MASK;
while(1)
{
    if(recflag==1)
    {
        project2();
    }
}
return 0;
}

// EOF
***** */
*** Authors: Harsimrarsingh Bindra, Smitesh Modak
*** Date Edited: 10/01/2017
**
*** File: conversion.c
**
*** Description: source file for data manipulation

```

```

* * -my_itoa
* * -my_atoi
* * -big_to_little32
* * -little_to_big32
*
*
* *
* ****

```

```

#include <stdint.h>
#include <stdio.h>
#include "conversion.h"           //invokes user defined header file for conversion
uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base) //function to convert integers to ascii strings
{
    uint8_t rem, i, temp, digits, count = 0;
    uint8_t sign = 0;
    uint8_t *last;
    if(data<0)                  //checks if data is negative and appends - to the string
    {
        data = -data;
        *ptr = '-';
        ptr++;
        count++;
        sign++;                  //to check the sign of
the input
    }
    if(data==0)
    {
        //to return zero
value if zero is entered
        *ptr='0';
    }
}

```

```

    digits=1;

    return digits;

}

while(data != 0)

{
    if((rem = data%base) > 9)           //condition if there is hex conversion

    {
        *ptr=('A'+ rem-10);           //adds the ascii value of 'A' character to
remainder

    }

    else

    {

        *ptr=rem + '0';             //for normal conversion adds the ascii
value of '0' to the remainder

    }

    data = data/base;

    ptr++;

    count++;

}

*ptr = '\0';

last = ptr-sizeof(uint8_t);

digits = count;

if(sign ==1)

{

    ptr = ptr - sizeof(uint8_t)*(count-1);

}

else

{

    ptr = ptr - sizeof(uint8_t)*(count);

}

```

```

}

if(count%2 !=0) //to check the string length whether its even or odd

{
    count =count-1;

}

for(i=0; i<(count/2); i++) //to reverse the string and store the ascii string in correct
order

{
    temp = *last;

    *last = *ptr;

    *ptr = temp;

    ptr++;

    last--;
}

ptr = ptr - sizeof(uint8_t)*(count/2);

return digits;
}

```

```

int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base)//function to convert ascii strings to integers

{
    if(ptr == NULL)

    {
        return Invalid_pointer;
    }

    int32_t integer=0;

    uint8_t i, sign=0;

    if(*ptr == '-')
        //to check whether the first character is '-' and set the sign
bit accordingly

    {

```

```

sign++;

ptr++;

digits = digits-1;

}

for(i = 0; i<digits; i++)           //to convert ascii strings to integers

{

if('0' <= *ptr && *ptr <='9')

{

integer = integer*base + *ptr - '0';      //subtracts the

ptr++;

}

else

{

integer = integer*base + *ptr -55;

ptr++;

}

}

if (sign == 1)           // to assign the proper sign to the integer according to the

sign value

{

integer = -integer;

}

return integer;

}

int8_t big_to_little32(uint32_t *data, uint32_t length) //to change the endianness from big to little

{

if(data == NULL)

{

```

```

    return Invalid_pointer;
}

    uint32_t temp, temp1, temp2, temp3;

    temp = *data & 0XFF000000;           //masking all but msb

    temp1 = *data & 0X000000FF;
    temp2 = *data & 0X00FF0000;
    temp3 = *data & 0X0000FF00;

    temp = temp >> 24;                //shifting msb to lsb

    temp1 = temp1<<24;
    temp2 = temp2>>8;
    temp3 = temp3<<8;

    *data = temp | temp1 | temp2 | temp3;      //assembles all bytes after proper shifting
and getting the required endianness

    return Pass;
}

```

```

int8_t little_to_big32(uint32_t *data, uint32_t length) //to change the endianness from little to big
{
    if(data == NULL)

    {
        return Invalid_pointer;
    }

    uint32_t temp, temp1, temp2, temp3;

    temp = *data & 0XFF000000;           //masking the lsb

    temp1 = *data & 0X000000FF;
    temp2 = *data & 0X00FF0000;
    temp3 = *data & 0X0000FF00;

    temp = temp >> 24;                //shifting the lsb

    temp1 = temp1<<24;

```

```

temp2 = temp2>>8;
temp3 = temp3<<8;

*data = temp | temp1 | temp2 | temp3;           //assembles all the bytes after proper
shifting to get the required endianness

return Pass;

}

```

```

#include <stdint.h>
#include<stdlib.h>
#include "circbuf.h"
#include "uart.h"
#include "conversion.h"

extern CB_t *recbuf;
extern uint8_t recflag;
uint32_t alpha_Count = 0;
uint32_t num_Count = 0;
uint32_t punc_Count = 0;
uint32_t misc_Count = 0;
uint32_t rec_DataCount = 0;

void project2()
{
    uint8_t rec_data=0;
    if(recbuf->count == 0)

```

```

{
    return Empty;
}

else
{
    rec_data = *(recbuf->head);

    if((rec_data>='a' && rec_data<='z') || (rec_data>='A' && rec_data<='Z'))
    {
        alpha_Count++;

        rec_DataCount++;
    }

    else if(rec_data>='0' && rec_data<='9')
    {
        num_Count++;

        rec_DataCount++;
    }

    else if (rec_data == '.' || rec_data == ',' || rec_data == '\"' || rec_data == ':' || rec_data
== ';' || rec_data == '!' || rec_data == "'")
    {
        punc_Count++;

        rec_DataCount++;
    }

    else
    {
        misc_Count++;

        rec_DataCount++;
    }
}

CB_buffer_remove_item(recbuf);

```

```

if(rec_DataCount == 10 || rec_data == 13)

{
    uint8_t *data = NULL;

    data = malloc(sizeof(uint32_t));

    uint32_t length = 0;

    uint8_t alphastring[]= "\n\rNo of Alphabets:";

    UART_send_n(alphastring, 20);

    length=my_itoa(alpha_Count, data, 10);

    UART_send_n(data, length);

    uint8_t Numstring[]= "\n\rNo of Numbers:";

    UART_send_n(Numstring, 18);

    length = my_itoa(num_Count, data, 10);

    UART_send_n(data, length);

    uint8_t Punstring[]= "\n\rNo of puncuations:";

    UART_send_n(Punstring, 22);

    length=my_itoa(punc_Count, data, 10);

    UART_send_n(data, length);

    uint8_t miscstring[]= "\n\rNo of Miscellaneous characters:";

    UART_send_n(miscstring, 35);

    length=my_itoa(misc_Count, data, 10);

    UART_send_n(data, length);

    rec_DataCount =0;

    free(data);

}

recflag = 0;

```

```
}
```

```
*****  
* * Authors: Harsimransingh Bindra, Smitesh Modak  
* * Date Edited: 10/01/2017  
* *  
* * File: memory.c  
* *  
* * Description: source file for data manipulation  
* * -my_memmove  
* * -my_memcpy  
* * -my_memset  
* * -my_memzero  
* * -my_reverse  
* * -reserve_words  
* * -free_words  
* *  
* *  
*****/
```

```
#include <stdint.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include "memory.h"
```

```

uint8_t my_memmove(uint8_t *src, uint8_t *dst, size_t length) //function to move data from one
memory location to another

{
    if(src == NULL || dst == NULL)

    {
        return Invalid_pointer;
    }

    uint8_t i = 0;

    if((src+sizeof(uint8_t)*length)>=dst && src<dst)      //Checks for overlapping of memory location

    {
        src = src+(sizeof(uint8_t)*(length-1));
        dst = dst+(sizeof(uint8_t)*(length-1));

        for(i = 0; i<length; i++)                         //to move data to the destination memory location

        {
            *dst = *src;

            src--;
            dst--;
        }
    }

    return Pass;
}

else
{
    for(i = 0; i<length; i++)
    {
        *dst = *src;

        src++;
        dst++;
    }
}

```

```

return Pass;
}

uint8_t my_memcpy(uint8_t *src, uint8_t *dst, size_t length) //function to copy data from one memory
location to another

{
    if(src == NULL || dst == NULL)
    {
        return Invalid_pointer;
    }

    uint8_t i = 0;

    for(i = 0; i<length; i++)           //to copy data from source memory address to destination
memory location

    {
        *dst = *src;

        src++;
        dst++;
    }

    return Pass;
}

```

```

uint8_t my_memset(uint8_t *src, size_t length, uint8_t value)//to set all given memory locations with a
particular value

{
    if(src == NULL)
    {
        return Invalid_pointer;
    }

    size_t i = length;

    while(i != 0)           //to set a particular value

```

```

    {
        *src = value;
        i--;
        src++;
    }

    src = src-(sizeof(uint8_t)*length);
    return Pass;
}

uint8_t my_memzero(uint8_t *src, size_t length)      //function to set all values at given memory
locations to zero

{
    if(src == NULL)
    {
        return Invalid_pointer;
    }

    size_t i = length;
    while(i != 0)
    {
        *src = 0;                      //sets all values to zero
        i--;
        src++;
    }

    src = src-(sizeof(uint8_t)*length);
    return Pass;
}

uint8_t my_reverse(uint8_t *src, size_t length)      //function to reverse a string

```

```

{

if(src == NULL)

{

return Invalid_pointer;

}

uint8_t *last;

    uint8_t temp,i;

    last = src + sizeof(uint8_t)*length-1;

    if(length%2 != 0)                                //to check whether the length of the string or number of
values are even or odd

    {

        length = length-1;

    }

for(i = 0; i<(length/2); i++)                  //to reverse the string at given memory location

    {

        temp = *src;

        *src = *last;

        *last = temp;

        src++;

        last--;

    }

src = src - (sizeof(uint8_t)*(length/2));

return Pass;

}

uint32_t *reserve_words(size_t length)

{

uint32_t *reserve = malloc(sizeof(uint32_t)*length);

return reserve;
}

```

```
}
```

```
void free_words(uint32_t *src) //function to free the dynamically allocated memory
{
    free(src);
}
```

```
/*
```

```
** #####
```

```
** Processors: MKL25Z128FM4
```

```
** MKL25Z128FT4
```

```
** MKL25Z128LH4
```

```
** MKL25Z128VLK4
```

```
**
```

```
** Compilers: Keil ARM C/C++ Compiler
```

```
** Freescale C/C++ for Embedded ARM
```

```
** GNU C Compiler
```

```
** GNU C Compiler - CodeSourcery Sourcery G++
```

```
** IAR ANSI C/C++ Compiler for ARM
```

```
**
```

```
** Reference manual: KL25P80M48SFORM, Rev.3, Sep 2012
```

```
** Version: rev. 2.5, 2015-02-19
```

```
** Build: b150220
```

```
**
```

```
** Abstract:
```

** Provides a system configuration function and a global variable that
** contains the system frequency. It configures the device and initializes
** the oscillator (PLL) that is part of the microcontroller device.
**
** Copyright (c) 2015 Freescale Semiconductor, Inc.
** All rights reserved.
**
** Redistribution and use in source and binary forms, with or without modification,
** are permitted provided that the following conditions are met:
**
** o Redistributions of source code must retain the above copyright notice, this list
** of conditions and the following disclaimer.
**
** o 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.
**
** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com

** mail: support@freescale.com

**

** Revisions:

** - rev. 1.0 (2012-06-13)

** Initial version.

** - rev. 1.1 (2012-06-21)

** Update according to reference manual rev. 1.

** - rev. 1.2 (2012-08-01)

** Device type UARTLP changed to UART0.

** - rev. 1.3 (2012-10-04)

** Update according to reference manual rev. 3.

** - rev. 1.4 (2012-11-22)

** MCG module - bit LOLS in MCG_S register renamed to LOLS0.

** NV registers - bit EZPORT_DIS in NV_FOPT register removed.

** - rev. 1.5 (2013-04-05)

** Changed start of doxygen comment.

** - rev. 2.0 (2013-10-29)

** Register accessor macros added to the memory map.

** Symbols for Processor Expert memory map compatibility added to the memory map.

** Startup file for gcc has been updated according to CMSIS 3.2.

** System initialization updated.

** - rev. 2.1 (2014-07-16)

** Module access macro module_BASES replaced by module_BASE_PTRS.

** System initialization and startup updated.

** - rev. 2.2 (2014-08-22)

```
**      System initialization updated - default clock config changed.  
**      - rev. 2.3 (2014-08-28)  
**      Update of startup files - possibility to override DefaultISR added.  
**      - rev. 2.4 (2014-10-14)  
**      Interrupt INT_LPTimer renamed to INT_LPTMR0.  
**      - rev. 2.5 (2015-02-19)  
**      Renamed interrupt vector LLW to LLWU.  
**  
** #####  
*/
```

```
/*!  
 * @file MKL25Z4  
 * @version 2.5  
 * @date 2015-02-19  
 * @brief Device specific configuration file for MKL25Z4 (implementation file)  
 *  
 * Provides a system configuration function and a global variable that contains  
 * the system frequency. It configures the device and initializes the oscillator  
 * (PLL) that is part of the microcontroller device.  
 */
```

```
#include <stdint.h>  
#include "MKL25Z4.h"  
  
/* -----  
-- Core clock
```

```

----- */
uint32_t SystemCoreClock = DEFAULT_SYSTEM_CLOCK;

/*
----- --
-- SystemInit()
----- */

void SystemInit (void) {
#if (DISABLE_WDOG)
/* SIM_COPC: COPT=0,COPCLKS=0,COPW=0 */
SIM->COPC = (uint32_t)0x00u;
#endif /* (DISABLE_WDOG) */
#ifndef CLOCK_SETUP
if((RCM->SRS0 & RCM_SRS0_WAKEUP_MASK) != 0x00U)
{
    if((PMC->REGSC & PMC_REGSC_ACKISO_MASK) != 0x00U)
    {
        PMC->REGSC |= PMC_REGSC_ACKISO_MASK; /* Release hold with ACKISO: Only has an effect if
recovering from VLLSx.*/
    }
}
#endif /* CLOCK_SETUP */

/* Power mode protection initialization */
#ifndef SYSTEM_SMC_PMPROT_VALUE
SMC->PMPROT = SYSTEM_SMC_PMPROT_VALUE;
#endif /* SYSTEM_SMC_PMPROT_VALUE */

/* System clock initialization */

```

```

/* Internal reference clock trim initialization */

#if defined(SLOW_TRIM_ADDRESS)

    if ( *((uint8_t*)SLOW_TRIM_ADDRESS) != 0xFFU) { /* Skip if non-volatile flash memory
is erased */

        MCG->C3 = *((uint8_t*)SLOW_TRIM_ADDRESS);

    #endif /* defined(SLOW_TRIM_ADDRESS) */

    #if defined(SLOW_FINE_TRIM_ADDRESS)

        MCG->C4 = (MCG->C4 & ~(MCG_C4_SCFTRIM_MASK)) | ((*(uint8_t*) SLOW_FINE_TRIM_ADDRESS))
& MCG_C4_SCFTRIM_MASK);

    #endif

    #if defined(FAST_TRIM_ADDRESS)

        MCG->C4 = (MCG->C4 & ~(MCG_C4_FCTRIM_MASK)) |((*(uint8_t*) FAST_TRIM_ADDRESS)) &
MCG_C4_FCTRIM_MASK);

    #endif

    #if defined(SLOW_TRIM_ADDRESS)

    }

    #endif /* defined(SLOW_TRIM_ADDRESS) */

/* Set system prescalers and clock sources */

SIM->CLKDIV1 = SYSTEM_SIM_CLKDIV1_VALUE; /* Set system prescalers */

SIM->SOPT1 = ((SIM->SOPT1) & (uint32_t)(~(SIM_SOPT1_OSC32KSEL_MASK))) |
((SYSTEM_SIM_SOPT1_VALUE) & (SIM_SOPT1_OSC32KSEL_MASK)); /* Set 32 kHz clock source
(ERCLK32K) */

SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_PLLFLLSEL_MASK))) |
((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_PLLFLLSEL_MASK)); /* Selects the high frequency clock
for various peripheral clocking options. */

SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_TPMSRC_MASK))) |
((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_TPMSRC_MASK)); /* Selects the clock source for the TPM
counter clock. */

#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FBI) || (MCG_MODE ==
MCG_MODE_BLPI))

/* Set MCG and OSC */

```

```

#if (((SYSTEM_OSCO_CR_VALUE) & OSC_CR_ERCLKEN_MASK) != 0x00U) ||
((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKENO_MASK) != 0x00U))

/* SIM_SCGC5: PORTA=1 */

SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

/* PORTA_PCR18: ISF=0,MUX=0 */

PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFS0_MASK) != 0x00U) {

/* PORTA_PCR19: ISF=0,MUX=0 */

PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

}

#endif

MCG->SC = SYSTEM_MCG_SC_VALUE; /* Set SC (fast clock internal reference divider) */

MCG->C1 = SYSTEM_MCG_C1_VALUE; /* Set C1 (clock source selection, FLL ext. reference divider,
int. reference enable etc.) */

/* Check that the source of the FLL reference clock is the requested one. */

if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {

while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {

}

} else {

while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {

}

}

MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set C2 (freq. range, ext.
and int. reference selection etc.; low power bit is set later) */

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK | MCG_C4_SCFTRIM_MASK)); /*

Set C4 (FLL output; trim values not changed) */

OSCO->CR = SYSTEM_OSCO_CR_VALUE; /* Set OSC_CR (OSCRCLK enable, oscillator capacitor load)
*/
#endif (MCG_MODE == MCG_MODE_BLPI)

/* BLPI specific */

```

```

MCG->C2 |= (MCG_C2_LP_MASK);      /* Disable FLL and PLL in bypass mode */

#endif

#else /* MCG_MODE */

/* Set MCG and OSC */

/* SIM_SCGC5: PORTA=1 */

SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

/* PORTA_PCR18: ISF=0,MUX=0 */

PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFSD0_MASK) != 0x00U) {

/* PORTA_PCR19: ISF=0,MUX=0 */

PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK | PORT_PCR_MUX(0x07)));

}

MCG->SC = SYSTEM_MCG_SC_VALUE;    /* Set SC (fast clock internal reference divider) */

MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set C2 (freq. range, ext.
and int. reference selection etc.; low power bit is set later) */

OSCO->CR = SYSTEM_OSCO_CR_VALUE; /* Set OSC_CR (OSCERCLK enable, oscillator capacitor load)
*/
#endif

#if (MCG_MODE == MCG_MODE_PEE)

MCG->C1 = (SYSTEM_MCG_C1_VALUE) | MCG_C1_CLKS(0x02); /* Set C1 (clock source selection, FLL
ext. reference divider, int. reference enable etc.) - PBE mode*/

#else

MCG->C1 = SYSTEM_MCG_C1_VALUE;    /* Set C1 (clock source selection, FLL ext. reference divider,
int. reference enable etc.) */

#endif

if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFSD0_MASK) != 0x00U) {

while((MCG->S & MCG_S_OSCINIT0_MASK) == 0x00U) { /* Check that the oscillator is running */

}

}

/* Check that the source of the FLL reference clock is the requested one. */

```

```

if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {
    while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {
    }
} else {
    while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {
    }
}

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK | MCG_C4_SCFTRIM_MASK)); /* Set C4 (FLL output; trim values not changed) */

#endif /* MCG_MODE */

/* Common for all MCG modes */

/* PLL clock can be used to generate clock for some devices regardless of clock generator (MCGOUTCLK) mode. */

MCG->C5 = (SYSTEM_MCG_C5_VALUE) & (uint8_t)(~(MCG_C5_PLLCLKENO_MASK)); /* Set C5 (PLL settings, PLL reference divider etc.) */

MCG->C6 = (SYSTEM_MCG_C6_VALUE) & (uint8_t)~(MCG_C6_PLLS_MASK); /* Set C6 (PLL select, VCO divider etc.) */

if ((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKENO_MASK) {

    MCG->C5 |= MCG_C5_PLLCLKENO_MASK; /* PLL clock enable in mode other than PEE or PBE */

}

/* BLPE, PEE and PBE MCG mode specific */

#if (MCG_MODE == MCG_MODE_BLPE)

    MCG->C2 |= (MCG_C2_LP_MASK); /* Disable FLL and PLL in bypass mode */

#elif ((MCG_MODE == MCG_MODE_PBE) || (MCG_MODE == MCG_MODE_PEE))

    MCG->C6 |= (MCG_C6_PLLS_MASK); /* Set C6 (PLL select, VCO divider etc.) */

    while((MCG->S & MCG_S_LOCK0_MASK) == 0x00U) { /* Wait until PLL is locked*/

```

```

}

#if (MCG_MODE == MCG_MODE_PEE)

    MCG->C1 &= (uint8_t)~(MCG_C1_CLKS_MASK);

#endif

#endif

#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FEE))

    while((MCG->S & MCG_S_CLKST_MASK) != 0x00U) { /* Wait until output of the FLL is selected */

    }

    /* Use LPTMR to wait for 1ms for FLL clock stabilization */

    SIM_SCGC5 |= SIM_SCGC5_LPTMR_MASK; /* Allow software control of LPMTR */

    LPTMR0->CMR = LPTMR_CMR_COMPARE(0); /* Default 1 LPO tick */

    LPTMR0->CSR = (LPTMR_CSR_TCF_MASK | LPTMR_CSR_TPS(0x00));

    LPTMR0->PSR = (LPTMR_PSR_PCS(0x01) | LPTMR_PSR_PBYP_MASK); /* Clock source: LPO, Prescaler bypass enable */

    LPTMR0->CSR = LPTMR_CSR_TEN_MASK; /* LPMTR enable */

    while((LPTMR0_CSR & LPTMR_CSR_TCF_MASK) == 0u) {

    }

    LPTMR0_CSR = 0x00; /* Disable LPTMR */

    SIM_SCGC5 &= (uint32_t)~(uint32_t)SIM_SCGC5_LPTMR_MASK;

#elif ((MCG_MODE == MCG_MODE_FBI) || (MCG_MODE == MCG_MODE_BLPI))

    while((MCG->S & MCG_S_CLKST_MASK) != 0x04U) { /* Wait until internal reference clock is selected as MCG output */

    }

#elif ((MCG_MODE == MCG_MODE_FBE) || (MCG_MODE == MCG_MODE_PBE) || (MCG_MODE == MCG_MODE_BLPE))

    while((MCG->S & MCG_S_CLKST_MASK) != 0x08U) { /* Wait until external reference clock is selected as MCG output */

    }

#elif (MCG_MODE == MCG_MODE_PEE)

    while((MCG->S & MCG_S_CLKST_MASK) != 0x0CU) { /* Wait until output of the PLL is selected */

```

```

    }

#endif

#if (((SYSTEM_SMC_PMCTRL_VALUE) & SMC_PMCTRL_RUNM_MASK) == (0x02U <<
SMC_PMCTRL_RUNM_SHIFT))

    SMC->PMCTRL = (uint8_t)((SYSTEM_SMC_PMCTRL_VALUE) & (SMC_PMCTRL_RUNM_MASK)); /*

Enable VLPR mode */

    while(SMC->PMSTAT != 0x04U) {      /* Wait until the system is in VLPR mode */

    }

#endif

```

```

/* PLL loss of lock interrupt request initialization */

if (((SYSTEM_MCG_C6_VALUE) & MCG_C6_LOLIE0_MASK) != 0U) {

    NVIC_EnableIRQ(MCG_IRQn);      /* Enable PLL loss of lock interrupt request */

}

#endif
}

```

```

/*
-- SystemCoreClockUpdate()
----- */

```

```

void SystemCoreClockUpdate (void) {

    uint32_t MCGOUTClock;          /* Variable to store output clock frequency of the MCG module */

    uint16_t Divider;

    if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x00U) {

        /* Output of FLL or PLL is selected */

        if ((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U) {

            /* FLL is selected */

```

```

if ((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U) {
    /* External reference clock is selected */

    MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG clock */

    if ((MCG->C2 & MCG_C2_RANGE0_MASK) != 0x00U) {
        switch (MCG->C1 & MCG_C1_FRDIV_MASK) {

            case 0x38U:
                Divider = 1536U;
                break;

            case 0x30U:
                Divider = 1280U;
                break;

            default:
                Divider = (uint16_t)(32LU << ((MCG->C1 & MCG_C1_FRDIV_MASK) >> MCG_C1_FRDIV_SHIFT));
                break;
        }
    } else /* ((MCG->C2 & MCG_C2_RANGE_MASK) != 0x00U) */
        Divider = (uint16_t)(1LU << ((MCG->C1 & MCG_C1_FRDIV_MASK) >> MCG_C1_FRDIV_SHIFT));
    }

    MCGOUTClock = (MCGOUTClock / Divider); /* Calculate the divided FLL reference clock */

} else /* (!(MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U) */
    MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* The slow internal reference clock is selected */

/* (!(MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U) */

/* Select correct multiplier to calculate the MCG output clock */

switch (MCG->C4 & (MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) {

    case 0x00U:
        MCGOUTClock *= 640U;
        break;

    case 0x20U:
        MCGOUTClock *= 1280U;
}

```

```

break;

case 0x40U:
    MCGOUTClock *= 1920U;
    break;

case 0x60U:
    MCGOUTClock *= 2560U;
    break;

case 0x80U:
    MCGOUTClock *= 732U;
    break;

case 0xA0U:
    MCGOUTClock *= 1464U;
    break;

case 0xC0U:
    MCGOUTClock *= 2197U;
    break;

case 0xE0U:
    MCGOUTClock *= 2929U;
    break;

default:
    break;
}

} else { /* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */

/* PLL is selected */

    Divider = (((uint16_t)MCG->C5 & MCG_C5_PRDIV0_MASK) + 0x01U);
    MCGOUTClock = (uint32_t)(CPU_XTAL_CLK_HZ / Divider); /* Calculate the PLL reference clock */
    Divider = (((uint16_t)MCG->C6 & MCG_C6_VDIV0_MASK) + 24U);
    MCGOUTClock *= Divider;      /* Calculate the MCG output clock */
}

/* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */

```

```

} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x40U) {

/* Internal reference clock is selected */

if ((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U) {

MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* Slow internal reference clock selected */

} else { /* !(MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U) */

    Divider = (uint16_t)(0x01LU << ((MCG->SC & MCG_SC_FCRDIV_MASK) >> MCG_SC_FCRDIV_SHIFT));

    MCGOUTClock = (uint32_t) (CPU_INT_FAST_CLK_HZ / Divider); /* Fast internal reference clock
selected */

} /* !(MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U) */

} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U) {

/* External reference clock is selected */

MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG clock */

} else { /* !(MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U) */

/* Reserved value */

return;

} /* !(MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U) */

SystemCoreClock = (MCGOUTClock / (0x01U + ((SIM->CLKDIV1 & SIM_CLKDIV1_OUTDIV1_MASK) >>
SIM_CLKDIV1_OUTDIV1_SHIFT)));

}

/*********************//**/

* @file core_cmSimd.h

* @brief CMSIS Cortex-M SIMD Header File

* @version V4.10

* @date 18. March 2015

*

* @note

*

*****/

/* Copyright (c) 2009 - 2014 ARM LIMITED

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )
```

```
#pragma system_include /* treat file as system include file for MISRA check */
```

```

#endif

#ifndef __CORE_CMSIMD_H
#define __CORE_CMSIMD_H


#ifndef __cplusplus
extern "C" {

#endif


/********************* Hardware Abstraction Layer ********************/
/* ##### Compiler specific Intrinsics ###### */
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
 * Access to dedicated SIMD instructions
 @{
 */

#if defined (__CC_ARM) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#define __SADD8          __sadd8
#define __QADD8          __qadd8
#define __SHADD8         __shadd8
#define __UADD8          __uadd8
#define __UQADD8         __uqadd8
#define __UHADD8         __uhadd8

```

```
#define __SSUB8          __ssub8
#define __QSUB8          __qsub8
#define __SHSUB8          __shsub8
#define __USUB8           __usub8
#define __UQSUB8          __uqsub8
#define __UHSUB8          __uhsub8
#define __SADD16          __sadd16
#define __QADD16          __qadd16
#define __SHADD16          __shadd16
#define __UADD16          __uadd16
#define __UQADD16          __uqadd16
#define __UHADD16          __uhadd16
#define __SSUB16          __ssub16
#define __QSUB16          __qsub16
#define __SHSUB16          __shsub16
#define __USUB16           __usub16
#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX             __sasx
#define __QASX             __qasx
#define __SHASX            __shasx
#define __UASX             __uasx
#define __UQASX            __uqasx
#define __UHASX            __uhasx
#define __SSAX              __ssax
#define __QSAX              __qsax
#define __SHSAX             __shsax
#define __USAX              __usax
#define __UQSAX             __uqsax
```

```

#define __UHSAX           __uhsax
#define __USAD8            __usad8
#define __USADA8           __usada8
#define __SSAT16           __ssat16
#define __USAT16           __usat16
#define __UXTB16           __uxtb16
#define __UXTAB16          __uxtab16
#define __SXTB16           __sxtb16
#define __SXTAB16          __sxtab16
#define __SMUAD            __smuad
#define __SMUADX           __smuadx
#define __SMLAD             __smlad
#define __SMLADX           __smladx
#define __SMLALD            __smlald
#define __SMLALDX          __smlalidx
#define __SMUSD              __smusd
#define __SMUSDX           __smusdx
#define __SMLSD              __smlsd
#define __SMLSDX           __smlsdx
#define __SMLSLD             __smlsld
#define __SMLSIDX          __smlsldx
#define __SEL                __sel
#define __QADD              __qadd
#define __QSUB              __qsub

#define __PKHBT(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0x0000FFFFUL) | \
                                (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0xFFFF0000UL) | \

```

```

(((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL) )

#define __SMMLA(ARG1,ARG2,ARG3)      ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                         ((int64_t)(ARG3) << 32)    ) >> 32))

#endif /* _SMM_H */

#ifndef __ASSEMBLY__
#elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
}
#endif /* __ASSEMBLY__ */

```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB8(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB8(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB8(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB8(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD16(uint32_t op1, uint32_t op2)
{
```

```
uint32_t result;

__ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SASX(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QASX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHASX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UASX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __USAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UQSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __USAD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __USADA8(uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}
```

```
#define __SSAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
})
```

```

__RES; \
}

#define __USAT16(ARG1,ARG2) \
({          \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SXTB16(uint32_t op1)
{
    uint32_t result;

```

```
__ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));  
return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUAD (uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUADX (uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}
```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLAD (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLADX (uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLALD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
    (llr.w32[0]), "1" (llr.w32[1]) );
#endif
}

```

```

#else           // Big endian

    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
    (llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);

}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;

llr.w64 = acc;

#endifdef __ARMEB__ // Little endian

__ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else           // Big endian

    __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
    (llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);

}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMUSD (uint32_t op1, uint32_t op2)
{

```

```

    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUSDX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD (uint32_t op1, uint32_t op2,
    uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX (uint32_t op1, uint32_t op2,
    uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3) );
}

```

```

return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSLD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSLDX (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;

```

```

llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SEL (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __QADD(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB(uint32_t op1, uint32_t op2)

{
    uint32_t result;

    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

```

```

#define __PKHBT(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
    __RES; \
})

```

```

#define __PKHTB(ARG1,ARG2,ARG3) \
({           \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2) ); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
    __RES; \
})

```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMMLA (int32_t op1, int32_t op2, int32_t
op3)
{
    int32_t result;

```

```
__ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" (op2), "r" (op3) );
return(result);
}

#ifndef CMSIS_DSP_H
#define CMSIS_DSP_H

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING carm specific functions */
/* not yet supported */

#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD_intrinsics */
```

```
#ifdef __cplusplus
}

#endif /* __CORE_CMSIMD_H */
/*********************//**/

* @file core_cm4.h
* @brief CMSIS Cortex-M4 Core Peripheral Access Layer Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*
*****/
/* Copyright (c) 2009 - 2015 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )  
#pragma system_include /* treat file as system include file for MISRA check */  
#endif
```

```
#ifndef __CORE_CM4_H_GENERIC  
#define __CORE_CM4_H_GENERIC
```

```
#ifdef __cplusplus  
extern "C" {  
#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions  
CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.

*/

```
*****  
*      CMSIS definitions  
*****/  
/** \ingroup Cortex_M4  
{@  
*/  
  
/* CMSIS CM4 definitions */  
  
#define __CM4_CMSIS_VERSION_MAIN (0x04)          /*!< [31:16] CMSIS HAL main version  
*/  
  
#define __CM4_CMSIS_VERSION_SUB (0x00)           /*!< [15:0] CMSIS HAL sub version */  
  
#define __CM4_CMSIS_VERSION ((__CM4_CMSIS_VERSION_MAIN << 16) | \  
                           __CM4_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */  
  
#define __CORTEX_M      (0x04)          /*!< Cortex-M Core */  
  
#if defined (__CC_ARM)  
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */  
*/
```

```

#define __INLINE      __inline           /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#if defined ( __GNUC__ )

#define __ASM        __asm            /*!< asm keyword for GNU Compiler */
#define __INLINE     inline           /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#endif

#if defined ( __ICCARM__ )

#define __ASM        __asm            /*!< asm keyword for IAR Compiler */
#define __INLINE     inline           /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */

#define __STATIC_INLINE static inline

#endif

#if defined ( __TMS470__ )

#define __ASM        __asm            /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#endif

#if defined ( __TASKING__ )

#define __ASM        __asm            /*!< asm keyword for TASKING Compiler */
#define __INLINE     inline           /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

#endif

#if defined ( __CSMC__ )

#define __packed

#define __ASM        __asm            /*!< asm keyword for COSMIC Compiler */
#define __INLINE     inline           /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */

#define __STATIC_INLINE static inline

#endif

```

```
#endif
```

```
/** __FPU_USED indicates whether an FPU is used or not.
```

For this, __FPU_PRESENT has to be checked prior to making use of FPU specific registers and functions.

```
*/
```

```
#if defined ( __CC_ARM )
```

```
    #if defined __TARGET_FPU_VFP
```

```
        #if (__FPU_PRESENT == 1)
```

```
            #define __FPU_USED    1
```

```
        #else
```

```
            #warning "Compiler generates FPU instructions for a device without an FPU (check  
__FPU_PRESENT)"
```

```
            #define __FPU_USED    0
```

```
        #endif
```

```
    #else
```

```
        #define __FPU_USED    0
```

```
    #endif
```

```
#elif defined ( __GNUC__ )
```

```
    #if defined (__VFP_FP__) && !defined(__SOFTFP__)
```

```
        #if (__FPU_PRESENT == 1)
```

```
            #define __FPU_USED    1
```

```
        #else
```

```
            #warning "Compiler generates FPU instructions for a device without an FPU (check  
__FPU_PRESENT)"
```

```
            #define __FPU_USED    0
```

```
        #endif
```

```
    #else
```

```
#define __FPU_USED    0
#endif

#ifndef __ICCARM__
#define __FPU_USED    0
#endif

#define __FPU_USED    0
#endif

#define __FPU_USED    0
#endif

#define __FPU_USED    0
#endif
```

```

#ifndef __FPU_USED
#define __FPU_USED 0

#endif /* __FPU_USED */

/* Standard types definitions */
#include <stdint.h>
/* Core Instruction Access */
#include <core_cmInstr.h>
/* Core Function Access */
#include <core_cmFunc.h>
/* Compiler specific SIMD Intrinsics */
#include <core_cmSimd.h>

```

```
#ifdef __cplusplus
}

#endif /* __CORE_CM4_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM4_H_DEPENDANT
#define __CORE_CM4_H_DEPENDANT

#endif /* __cplusplus
extern "C" {

#endif /* __CORE_CM4_H_DEPENDANT

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
#ifndef __CM4_REV
#define __CM4_REV      0x0000
#warning "__CM4_REV not defined in device header file; using default!"
#endif /* __CM4_REV

#ifndef __FPU_PRESENT
#define __FPU_PRESENT 0
#warning "__FPU_PRESENT not defined in device header file; using default!"
#endif /* __FPU_PRESENT

#ifndef __MPU_PRESENT
```

```

#define __MPU_PRESENT      0
#warning "__MPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS     4
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"

#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"

#endif

#endif

/* IO definitions (access restrictions to peripheral registers) */
/** \defgroup CMSIS_glob_defs CMSIS Global Defines

<strong>IO Type Qualifiers</strong> are used
\li to specify the access to peripheral variables.
\li for automatic generation of peripheral register debug information.
*/
#endif

#ifndef __cplusplus
#define __I volatile      /*!< Defines 'read only' permissions      */
#else
#define __I volatile const /*!< Defines 'read only' permissions      */
#endif
#define __O volatile      /*!< Defines 'write only' permissions    */

```

```

#define __IO volatile /*!< Defines 'read / write' permissions */

/*}@} end of group Cortex_M4 */

/*********************  

*      Register Abstraction  

Core Register contain:  

- Core Register  

- Core NVIC Register  

- Core SCB Register  

- Core SysTick Register  

- Core Debug Register  

- Core MPU Register  

- Core FPU Register  

*****  

/** \defgroup CMSIS_core_register Defines and Type Definitions  

 \brief Type definitions and defines for Cortex-M processor based devices.  

 */  

  

/** \ingroup CMSIS_core_register  

 \defgroup CMSIS_CORE Status and Control Registers  

 \brief Core Register type definitions.  

@{  

*/  

  

/** \brief Union type to access the Application Program Status Register (APSR).  

 */

```

```

typedef union
{
    struct
    {
        uint32_t _reserved0:16;           /*!< bit: 0..15 Reserved          */
        uint32_t GE:4;                 /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:7;           /*!< bit: 20..26 Reserved          */
        uint32_t Q:1;                  /*!< bit: 27 Saturation condition flag */
        uint32_t V:1;                  /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;                  /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;                  /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;                  /*!< bit: 31 Negative condition code flag */
    } b;                         /*!< Structure used for bit access      */
    uint32_t w;                   /*!< Type     used for word access      */
} APSR_Type;

/* APSR Register Definitions */

#define APSR_N_Pos            31           /*!< APSR: N Position */
#define APSR_N_Msk             (1UL << APSR_N_Pos)      /*!< APSR: N Mask */

#define APSR_Z_Pos            30           /*!< APSR: Z Position */
#define APSR_Z_Msk             (1UL << APSR_Z_Pos)      /*!< APSR: Z Mask */

#define APSR_C_Pos            29           /*!< APSR: C Position */
#define APSR_C_Msk             (1UL << APSR_C_Pos)      /*!< APSR: C Mask */

#define APSR_V_Pos            28           /*!< APSR: V Position */
#define APSR_V_Msk             (1UL << APSR_V_Pos)      /*!< APSR: V Mask */

```

```

#define APSR_Q_Pos      27          /*!< APSR: Q Position */
#define APSR_Q_Msk      (1UL << APSR_Q_Pos)    /*!< APSR: Q Mask */

#define APSR_GE_Pos     16          /*!< APSR: GE Position */
#define APSR_GE_Msk     (0xFUL << APSR_GE_Pos)  /*!< APSR: GE Mask */

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

*/

```

typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0.. 8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} IPSR_Type;

```

/* IPSR Register Definitions */

```

#define IPSR_ISR_Pos      0          /*!< IPSR: ISR Position */
#define IPSR_ISR_Msk      (0x1FFUL /*<< IPSR_ISR_Pos*/)  /*!< IPSR: ISR Mask */

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).

*/

```

typedef union
{
    struct

```

```

{
    uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
    uint32_t _reserved0:7;     /*!< bit: 9..15 Reserved */
    uint32_t GE:4;            /*!< bit: 16..19 Greater than or Equal flags */
    uint32_t _reserved1:4;     /*!< bit: 20..23 Reserved */
    uint32_t T:1;              /*!< bit: 24 Thumb bit (read 0) */
    uint32_t IT:2;             /*!< bit: 25..26 saved IT state (read 0) */
    uint32_t Q:1;              /*!< bit: 27 Saturation condition flag */
    uint32_t V:1;              /*!< bit: 28 Overflow condition code flag */
    uint32_t C:1;              /*!< bit: 29 Carry condition code flag */
    uint32_t Z:1;              /*!< bit: 30 Zero condition code flag */
    uint32_t N:1;              /*!< bit: 31 Negative condition code flag */

} b;                      /*!< Structure used for bit access */

uint32_t w;                /*!< Type used for word access */

} xPSR_Type;

```

```

/* xPSR Register Definitions */

#define xPSR_N_Pos          31           /*!< xPSR: N Position */
#define xPSR_N_Msk          (1UL << xPSR_N_Pos)      /*!< xPSR: N Mask */

#define xPSR_Z_Pos          30           /*!< xPSR: Z Position */
#define xPSR_Z_Msk          (1UL << xPSR_Z_Pos)      /*!< xPSR: Z Mask */

#define xPSR_C_Pos          29           /*!< xPSR: C Position */
#define xPSR_C_Msk          (1UL << xPSR_C_Pos)      /*!< xPSR: C Mask */

#define xPSR_V_Pos          28           /*!< xPSR: V Position */
#define xPSR_V_Msk          (1UL << xPSR_V_Pos)      /*!< xPSR: V Mask */

```

```

#define xPSR_Q_Pos           27          /*!< xPSR: Q Position */
#define xPSR_Q_Msk          (1UL << xPSR_Q_Pos)    /*!< xPSR: Q Mask */

#define xPSR_IT_Pos          25          /*!< xPSR: IT Position */
#define xPSR_IT_Msk         (3UL << xPSR_IT_Pos)    /*!< xPSR: IT Mask */

#define xPSR_T_Pos            24          /*!< xPSR: T Position */
#define xPSR_T_Msk          (1UL << xPSR_T_Pos)    /*!< xPSR: T Mask */

#define xPSR_GE_Pos           16          /*!< xPSR: GE Position */
#define xPSR_GE_Msk         (0xFUL << xPSR_GE_Pos)  /*!< xPSR: GE Mask */

#define xPSR_ISR_Pos           0          /*!< xPSR: ISR Position */
#define xPSR_ISR_Msk        (0x1FFUL /*<< xPSR_ISR_Pos*/) /*!< xPSR: ISR Mask */

```

```

/** \brief Union type to access the Control Registers (CONTROL).

*/
typedef union
{
    struct
    {
        uint32_t nPRIV:1;           /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                         /*!< Structure used for bit access */
    uint32_t w;                  /*!< Type used for word access */
} CONTROL_Type;

```

```

/* CONTROL Register Definitions */

#define CONTROL_FPCA_Pos           2                  /*!< CONTROL: FPCA Position */
#define CONTROL_FPCA_Msk          (1UL << CONTROL_FPCA_Pos)      /*!< CONTROL: FPCA
Mask */

#define CONTROL_SPSEL_Pos          1                  /*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk         (1UL << CONTROL_SPSEL_Pos)      /*!< CONTROL: SPSEL
Mask */

#define CONTROL_nPRIV_Pos          0                  /*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk        (1UL /*<< CONTROL_nPRIV_Pos*/)      /*!< CONTROL:
nPRIV Mask */

/*@} end of group CMSIS_CORE */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers

@{

*/

```

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).

```

typedef struct
{
    __IO uint32_t ISER[8];           /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */
    uint32_t RESERVED0[24];
    __IO uint32_t ICER[8];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */

```

```

    uint32_t RSERVED1[24];

__IO uint32_t ISPR[8];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */

    uint32_t RESERVED2[24];

__IO uint32_t ICPR[8];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */

    uint32_t RESERVED3[24];

__IO uint32_t IABR[8];           /*!< Offset: 0x200 (R/W) Interrupt Active bit Register */

    uint32_t RESERVED4[56];

__IO uint8_t IP[240];           /*!< Offset: 0x300 (R/W) Interrupt Priority Register (8Bit wide) */

    uint32_t RESERVED5[644];

__O uint32_t STIR;              /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register */

} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */

#define NVIC_STIR_INTID_Pos      0                  /*!< STIR: INTLINESNUM Position */

#define NVIC_STIR_INTID_Msk      (0x1FUL /*<< NVIC_STIR_INTID_Pos*/)  /*!< STIR: INTLINESNUM Mask */

```

```
/*@} end of group CMSIS_NVIC */
```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_SCB System Control Block (SCB)
 \brief Type definitions for the System Control Block Registers

 @{
 */

```

```

/** \brief Structure type to access the System Control Block (SCB).
 */

```

```
typedef struct
```

```

{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
*/                          

    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
    __IO uint32_t AIRCR;         /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register   */
    __IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register          */
    __IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register    */
    __IO uint8_t SHP[12];        /*!< Offset: 0x018 (R/W) System Handlers Priority Registers (4-7, 8-11,
12-15) */
    __IO uint32_t SHCSR;         /*!< Offset: 0x024 (R/W) System Handler Control and State Register */
*/                          

    __IO uint32_t CFSR;          /*!< Offset: 0x028 (R/W) Configurable Fault Status Register */
*/                          

    __IO uint32_t HFSR;          /*!< Offset: 0x02C (R/W) HardFault Status Register          */
    __IO uint32_t DFSR;          /*!< Offset: 0x030 (R/W) Debug Fault Status Register        */
    __IO uint32_t MMFAR;         /*!< Offset: 0x034 (R/W) MemManage Fault Address Register    */
*/                          

    __IO uint32_t BFAR;          /*!< Offset: 0x038 (R/W) BusFault Address Register          */
    __IO uint32_t AFSR;          /*!< Offset: 0x03C (R/W) Auxiliary Fault Status Register    */
    __I uint32_t PFR[2];         /*!< Offset: 0x040 (R/ ) Processor Feature Register        */
    __I uint32_t DFR;           /*!< Offset: 0x048 (R/ ) Debug Feature Register          */
    __I uint32_t ADR;            /*!< Offset: 0x04C (R/ ) Auxiliary Feature Register        */
    __I uint32_t MMFR[4];        /*!< Offset: 0x050 (R/ ) Memory Model Feature Register    */
*/                          

    __I uint32_t ISAR[5];        /*!< Offset: 0x060 (R/ ) Instruction Set Attributes Register */
    uint32_t RESERVED0[5];       /* */

    __IO uint32_t CPACR;         /*!< Offset: 0x088 (R/W) Coprocessor Access Control Register */
*/
}

} SCB_Type;

```

```

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24           /*!< SCB CPUID:  
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk     (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)    /*!<  
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos        20           /*!< SCB CPUID: VARIANT Position  
*/

#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos)    /*!< SCB CPUID:  
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos   16           /*!< SCB CPUID:  
ARCHITECTURE Position */

#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos)    /*!<  
SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos         4            /*!< SCB CPUID: PARTNO Position  
*/

#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos)    /*!< SCB  
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos       0            /*!< SCB CPUID: REVISION Position  
*/

#define SCB_CPUID_REVISION_Msk      (0xFUL /*<< SCB_CPUID_REVISION_Pos*/)    /*!< SCB  
CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos     31           /*!< SCB ICSR: NMIPENDSET  
Position */

#define SCB_ICSR_NMIPENDSET_Msk     (1UL << SCB_ICSR_NMIPENDSET_Pos)    /*!< SCB ICSR:  
NMIPENDSET Mask */

```

#define SCB_ICSR_PENDSVSET_Pos	28	/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk	(1UL << SCB_ICSR_PENDSVSET_Pos)	/*!< SCB ICSR: PENDSVSET Mask */
#define SCB_ICSR_PENDSVCLR_Pos	27	/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */
#define SCB_ICSR_VECTPENDING_Pos	12	/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk	(0x1FFUL << SCB_ICSR_VECTPENDING_Pos)	/*!< SCB ICSR: VECTPENDING Mask */

```

#define SCB_ICSR_RETTOBASE_Pos      11          /*!< SCB ICSR: RETTOBASE
Position */

#define SCB_ICSR_RETTOBASE_Msk      (1UL << SCB_ICSR_RETTOBASE_Pos)      /*!< SCB ICSR:
RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos     0           /*!< SCB ICSR: VECTACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk     (0x1FFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/)  /*!< SCB
ICSR: VECTACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */

#define SCB_VTOR_TBLOFF_Pos        7           /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk        (0xFFFFFFFFUL << SCB_VTOR_TBLOFF_Pos)      /*!< SCB
VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos      16          /*!< SCB AIRCR: VECTKEY Position */

#define SCB_AIRCR_VECTKEY_Msk      (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)      /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos   16          /*!< SCB AIRCR: VECTKEYSTAT
Position */

#define SCB_AIRCR_VECTKEYSTAT_Msk   (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)      /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos    15          /*!< SCB AIRCR: ENDIANESS
Position */

#define SCB_AIRCR_ENDIANESS_Msk    (1UL << SCB_AIRCR_ENDIANESS_Pos)      /*!< SCB
AIRCR: ENDIANESS Mask */

```

```

#define SCB_AIRCR_PRIGROUP_Pos      8          /*!< SCB AIRCR: PRIGROUP
Position */

#define SCB_AIRCR_PRIGROUP_Msk     (7UL << SCB_AIRCR_PRIGROUP_Pos)    /*!< SCB
AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos   2          /*!< SCB AIRCR: SYSRESETREQ
Position */

#define SCB_AIRCR_SYSRESETREQ_Msk   (1UL << SCB_AIRCR_SYSRESETREQ_Pos)  /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos  1          /*!< SCB AIRCR: VECTCLRACTIVE
Position */

#define SCB_AIRCR_VECTCLRACTIVE_Msk  (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos) /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

#define SCB_AIRCR_VECTRESET_Pos     0          /*!< SCB AIRCR: VECTRESET
Position */

#define SCB_AIRCR_VECTRESET_Msk     (1UL /*<< SCB_AIRCR_VECTRESET_Pos*/) /*!< SCB
AIRCR: VECTRESET Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos       4          /*!< SCB SCR: SEVONPEND
Position */

#define SCB_SCR_SEVONPEND_Msk       (1UL << SCB_SCR_SEVONPEND_Pos)    /*!< SCB SCR:
SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos       2          /*!< SCB SCR: SLEEPDEEP Position
*/

#define SCB_SCR_SLEEPDEEP_Msk       (1UL << SCB_SCR_SLEEPDEEP_Pos)    /*!< SCB SCR:
SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos     1          /*!< SCB SCR: SLEEPONEXIT
Position */

```

```

#define SCB_SCR_SLEEPONEXIT_Msk      (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR:
SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos        9                                /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk        (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR:
STKALIGN Mask */

#define SCB_CCR_BFHFMIGN_Pos        8                                /*!< SCB CCR: BFHFMIGN
Position */
#define SCB_CCR_BFHFMIGN_Msk        (1UL << SCB_CCR_BFHFMIGN_Pos)      /*!< SCB CCR:
BFHFMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos        4                                /*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk        (1UL << SCB_CCR_DIV_0_TRP_Pos)      /*!< SCB CCR:
DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos      3                                /*!< SCB CCR: UNALIGN_TRP
Position */
#define SCB_CCR_UNALIGN_TRP_Msk      (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB
CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos     1                                /*!< SCB CCR: USERSETMPEND
Position */
#define SCB_CCR_USERSETMPEND_Msk     (1UL << SCB_CCR_USERSETMPEND_Pos)      /*!< SCB
CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos   0                                /*!< SCB CCR:
NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDENA_Msk   (1UL /*<< SCB_CCR_NONBASETHRDENA_Pos*/) /*!< SCB CCR:
NONBASETHRDENA Mask */

```

```

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_USGFAULTENA_Pos      18          /*!< SCB SHCSR:  
USGFAULTENA Position */

#define SCB_SHCSR_USGFAULTENA_Msk     (1UL << SCB_SHCSR_USGFAULTENA_Pos)    /*!< SCB  
SHCSR: USGFAULTENA Mask */

#define SCB_SHCSR_BUSFAULTENA_Pos    17          /*!< SCB SHCSR: BUSFAULTENA  
Position */

#define SCB_SHCSR_BUSFAULTENA_Msk   (1UL << SCB_SHCSR_BUSFAULTENA_Pos)    /*!< SCB  
SHCSR: BUSFAULTENA Mask */

#define SCB_SHCSR_MEMFAULTENA_Pos    16          /*!< SCB SHCSR:  
MEMFAULTENA Position */

#define SCB_SHCSR_MEMFAULTENA_Msk   (1UL << SCB_SHCSR_MEMFAULTENA_Pos)    /*!<  
SCB SHCSR: MEMFAULTENA Mask */

#define SCB_SHCSR_SVCALLPENDED_Pos   15          /*!< SCB SHCSR:  
SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk  (1UL << SCB_SHCSR_SVCALLPENDED_Pos)    /*!< SCB  
SHCSR: SVCALLPENDED Mask */

#define SCB_SHCSR_BUSFAULTPENDED_Pos  14          /*!< SCB SHCSR:  
BUSFAULTPENDED Position */

#define SCB_SHCSR_BUSFAULTPENDED_Msk (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)    /*!<  
SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos  13          /*!< SCB SHCSR:  
MEMFAULTPENDED Position */

#define SCB_SHCSR_MEMFAULTPENDED_Msk (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)    /*!<  
SCB SHCSR: MEMFAULTPENDED Mask */

#define SCB_SHCSR_USGFAULTPENDED_Pos  12          /*!< SCB SHCSR:  
USGFAULTPENDED Position */

```

```

#define SCB_SHCSR_USGFAULTPENDED_Msk      (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)      /*!<
SCB SHCSR: USGFAULTPENDED Mask */

#define SCB_SHCSR_SYSTICKACT_Pos          11                                /*!< SCB SHCSR: SYSTICKACT
Position */

#define SCB_SHCSR_SYSTICKACT_Msk         (1UL << SCB_SHCSR_SYSTICKACT_Pos)      /*!< SCB
SHCSR: SYSTICKACT Mask */

#define SCB_SHCSR_PENDSVACT_Pos          10                                /*!< SCB SHCSR: PENDSVACT
Position */

#define SCB_SHCSR_PENDSVACT_Msk         (1UL << SCB_SHCSR_PENDSVACT_Pos)      /*!< SCB
SHCSR: PENDSVACT Mask */

#define SCB_SHCSR_MONITORACT_Pos          8                                 /*!< SCB SHCSR: MONITORACT
Position */

#define SCB_SHCSR_MONITORACT_Msk        (1UL << SCB_SHCSR_MONITORACT_Pos)     /*!< SCB
SHCSR: MONITORACT Mask */

#define SCB_SHCSR_SVCALLACT_Pos          7                                 /*!< SCB SHCSR: SVCALLACT
Position */

#define SCB_SHCSR_SVCALLACT_Msk        (1UL << SCB_SHCSR_SVCALLACT_Pos)     /*!< SCB
SHCSR: SVCALLACT Mask */

#define SCB_SHCSR_USGFAULTACT_Pos        3                                 /*!< SCB SHCSR: USGFAULTACT
Position */

#define SCB_SHCSR_USGFAULTACT_Msk       (1UL << SCB_SHCSR_USGFAULTACT_Pos)    /*!< SCB
SHCSR: USGFAULTACT Mask */

#define SCB_SHCSR_BUSFAULTACT_Pos        1                                 /*!< SCB SHCSR: BUSFAULTACT
Position */

#define SCB_SHCSR_BUSFAULTACT_Msk       (1UL << SCB_SHCSR_BUSFAULTACT_Pos)   /*!< SCB
SHCSR: BUSFAULTACT Mask */

```

```

#define SCB_SHCSR_MEMFAULTACT_Pos      0                  /*!< SCB SHCSR:  

MEMFAULTACT Position */  
  

#define SCB_SHCSR_MEMFAULTACT_Msk     (1UL /*<< SCB_SHCSR_MEMFAULTACT_Pos*/)    /*!<  

SCB SHCSR: MEMFAULTACT Mask */  
  

/* SCB Configurable Fault Status Registers Definitions */  
  

#define SCB_CFSR_USGFAULTSR_Pos      16                 /*!< SCB CFSR: Usage Fault  

Status Register Position */  
  

#define SCB_CFSR_USGFAULTSR_Msk      (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos)    /*!< SCB  

CFSR: Usage Fault Status Register Mask */  
  

#define SCB_CFSR_BUSFAULTSR_Pos      8                   /*!< SCB CFSR: Bus Fault Status  

Register Position */  
  

#define SCB_CFSR_BUSFAULTSR_Msk      (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos)    /*!< SCB  

CFSR: Bus Fault Status Register Mask */  
  

#define SCB_CFSR_MEMFAULTSR_Pos      0                  /*!< SCB CFSR: Memory Manage  

Fault Status Register Position */  
  

#define SCB_CFSR_MEMFAULTSR_Msk      (0xFFUL /*<< SCB_CFSR_MEMFAULTSR_Pos*/)    /*!<  

SCB CFSR: Memory Manage Fault Status Register Mask */  
  

/* SCB Hard Fault Status Registers Definitions */  
  

#define SCB_HFSR_DEBUGEVT_Pos        31                 /*!< SCB HFSR: DEBUGEVT  

Position */  
  

#define SCB_HFSR_DEBUGEVT_Msk        (1UL << SCB_HFSR_DEBUGEVT_Pos)    /*!< SCB HFSR:  

DEBUGEVT Mask */  
  

#define SCB_HFSR_FORCED_Pos         30                 /*!< SCB HFSR: FORCED Position */  
  

#define SCB_HFSR_FORCED_Msk         (1UL << SCB_HFSR_FORCED_Pos)    /*!< SCB HFSR:  

FORCED Mask */  
  

#define SCB_HFSR_VECTTBL_Pos        1                  /*!< SCB HFSR: VECTTBL Position */

```

```

#define SCB_HFSR_VECTTBL_Msk      (1UL << SCB_HFSR_VECTTBL_Pos)      /*!< SCB HFSR:
VECTTBL Mask */

/* SCB Debug Fault Status Register Definitions */

#define SCB_DFSR_EXTERNAL_Pos     4                                /*!< SCB DFSR: EXTERNAL Position
*/
#define SCB_DFSR_EXTERNAL_Msk    (1UL << SCB_DFSR_EXTERNAL_Pos)      /*!< SCB DFSR:
EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos       3                                /*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk     (1UL << SCB_DFSR_VCATCH_Pos)      /*!< SCB DFSR:
VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos      2                                /*!< SCB DFSR: DWTTRAP Position
*/
#define SCB_DFSR_DWTTRAP_Msk    (1UL << SCB_DFSR_DWTTRAP_Pos)      /*!< SCB DFSR:
DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos         1                                /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk       (1UL << SCB_DFSR_BKPT_Pos)      /*!< SCB DFSR: BKPT
Mask */

#define SCB_DFSR_HALTED_Pos       0                                /*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk     (1UL /*<< SCB_DFSR_HALTED_Pos*/)  /*!< SCB DFSR:
HALTED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
\brief Type definitions for the System Control and ID Register not in the SCB

```

```

@{
}

/** \brief Structure type to access the System Control and ID Register not in the SCB.
 */
typedef struct
{
    uint32_t RESERVED0[1];

    __I uint32_t ICTR;           /*!< Offset: 0x004 (R/ ) Interrupt Controller Type Register */
    __IO uint32_t ACTLR;         /*!< Offset: 0x008 (R/W) Auxiliary Control Register */
} SCnSCB_Type;

/* Interrupt Controller Type Register Definitions */

#define SCnSCB_ICTR_INTLINESNUM_Pos     0           /*!< ICTR: INTLINESNUM Position */
#define SCnSCB_ICTR_INTLINESNUM_Msk      (0xFUL /*<< SCnSCB_ICTR_INTLINESNUM_Pos*/) /*!<
ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define SCnSCB_ACTLR_DISOOFP_Pos       9           /*!< ACTLR: DISOOFP Position */
#define SCnSCB_ACTLR_DISOOFP_Msk        (1UL << SCnSCB_ACTLR_DISOOFP_Pos) /*!< ACTLR:
DISOOFP Mask */

#define SCnSCB_ACTLR_DISFPCA_Pos       8           /*!< ACTLR: DISFPCA Position */
#define SCnSCB_ACTLR_DISFPCA_Msk        (1UL << SCnSCB_ACTLR_DISFPCA_Pos) /*!< ACTLR:
DISFPCA Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos       2           /*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk        (1UL << SCnSCB_ACTLR_DISFOLD_Pos) /*!< ACTLR:
DISFOLD Mask */

```

```

#define SCnSCB_ACTLR_DISDEFWBUF_Pos      1           /*!< ACTLR: DISDEFWBUF
Position */

#define SCnSCB_ACTLR_DISDEFWBUF_Msk     (1UL << SCnSCB_ACTLR_DISDEFWBUF_Pos)    /*!<
ACTLR: DISDEFWBUF Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos    0           /*!< ACTLR: DISMCYCINT
Position */

#define SCnSCB_ACTLR_DISMCYCINT_Msk   (1UL /*<< SCnSCB_ACTLR_DISMCYCINT_Pos*/) /*!<
ACTLR: DISMCYCINT Mask */

/*@} end of group CMSIS_SCnotSCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SysTick System Tick Timer (SysTick)
\brief Type definitions for the System Timer Registers.

@{

*/

```

/** \brief Structure type to access the System Timer (SysTick).

```

*/

```

typedef struct

```

{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;         /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;          /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I uint32_t CALIB;        /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
}
```

} SysTick_Type;

```

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos      16          /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk     (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2           /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk    (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos      1           /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk     (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos       0           /*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk      (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)      /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos      0           /*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/)  /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos      0           /*!< SysTick VAL: CURRENT Position */
#define SysTick_VAL_CURRENT_Msk     (0xFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/)  /*!< SysTick VAL: CURRENT Mask */

```

```

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos      31           /*!< SysTick CALIB: NOREF Position */
/*/

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)      /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos      30           /*!< SysTick CALIB: SKEW Position */
/*/

#define SysTick_CALIB_SKEW_Msk     (1UL << SysTick_CALIB_SKEW_Pos)      /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos     0            /*!< SysTick CALIB: TENMS Position */
/*/

#define SysTick_CALIB_TENMS_Msk    (0xFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_ITM  Instrumentation Trace Macrocell (ITM)
\brief Type definitions for the Instrumentation Trace Macrocell (ITM)

@{
*/

```

```
/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
*/

```

```

typedef struct
{
    __O union
{

```

```

__O uint8_t u8;           /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit      */
__O uint16_t u16;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit     */
__O uint32_t u32;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit     */
} PORT [32];              /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers   */

uint32_t RESERVED0[864];

__IO uint32_t TER;         /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register    */
uint32_t RESERVED1[15];

__IO uint32_t TPR;         /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register */
uint32_t RESERVED2[15];

__IO uint32_t TCR;         /*!< Offset: 0xE80 (R/W) ITM Trace Control Register   */
uint32_t RESERVED3[29];

__O uint32_t IWR;          /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register */
__I uint32_t IRR;          /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register */
__IO uint32_t IMCR;         /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
uint32_t RESERVED4[43];

__O uint32_t LAR;          /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register     */
__I uint32_t LSR;          /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register     */
uint32_t RESERVED5[6];

__I uint32_t PID4;          /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
__I uint32_t PID5;          /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
__I uint32_t PID6;          /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
__I uint32_t PID7;          /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
__I uint32_t PID0;          /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
__I uint32_t PID1;          /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
__I uint32_t PID2;          /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
__I uint32_t PID3;          /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
__I uint32_t CID0;          /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
__I uint32_t CID1;          /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
__I uint32_t CID2;          /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */

```

```

__I uint32_t CID3;           /*!< Offset: 0xFFC (R/ ) ITM Component Identification Register #3 */

} ITM_Type;

/* ITM Trace Privilege Register Definitions */

#define ITM_TPR_PRIVMASK_Pos      0          /*!< ITM TPR: PRIVMASK Position */
*/
#define ITM_TPR_PRIVMASK_Msk      (0xFUL /*<< ITM_TPR_PRIVMASK_Pos*/)      /*!< ITM TPR: PRIVMASK Mask */

/* ITM Trace Control Register Definitions */

#define ITM_TCR_BUSY_Pos         23         /*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk         (1UL << ITM_TCR_BUSY_Pos)      /*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos   16         /*!< ITM TCR: ATBID Position */
#define ITM_TCR_TraceBusID_Msk   (0x7FUL << ITM_TCR_TraceBusID_Pos)      /*!< ITM TCR: ATBID Mask */

#define ITM_TCR_GTSFREQ_Pos      10         /*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk      (3UL << ITM_TCR_GTSFREQ_Pos)      /*!< ITM TCR: Global timestamp frequency Mask */

#define ITM_TCR_TSPrescale_Pos   8          /*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk   (3UL << ITM_TCR_TSPrescale_Pos)      /*!< ITM TCR: TSPrescale Mask */

#define ITM_TCR_SWOENA_Pos       4          /*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk       (1UL << ITM_TCR_SWOENA_Pos)      /*!< ITM TCR: SWOENA Mask */

#define ITM_TCR_DWTENA_Pos       3          /*!< ITM TCR: DWTENA Position */

```

```

#define ITM_TCR_DWTENA_Msk      (1UL << ITM_TCR_DWTENA_Pos)      /*!< ITM TCR:
DWTENA Mask */

#define ITM_TCR_SYNCENA_Pos     2                                /*!< ITM TCR: SYNCENA Position */
#define ITM_TCR_SYNCENA_Msk     (1UL << ITM_TCR_SYNCENA_Pos)      /*!< ITM TCR:
SYNCENA Mask */

#define ITM_TCR_TSENA_Pos       1                                /*!< ITM TCR: TSENA Position */
#define ITM_TCR_TSENA_Msk       (1UL << ITM_TCR_TSENA_Pos)      /*!< ITM TCR: TSENA
Mask */

#define ITM_TCR_ITMENA_Pos      0                                /*!< ITM TCR: ITM Enable bit
Position */
#define ITM_TCR_ITMENA_Msk     (1UL /*<< ITM_TCR_ITMENA_Pos*/) /*!< ITM TCR:
ITM Enable bit Mask */

/* ITM Integration Write Register Definitions */

#define ITM_IWR_ATVALIDIM_Pos   0                                /*!< ITM IWR: ATVALIDIM Position
*/
#define ITM_IWR_ATVALIDIM_Msk   (1UL /*<< ITM_IWR_ATVALIDIM_Pos*/) /*!< ITM IWR:
ATVALIDIM Mask */

/* ITM Integration Read Register Definitions */

#define ITMIRR_ATREADYM_Pos    0                                /*!< ITM IRR: ATREADYM Position
*/
#define ITMIRR_ATREADYM_Msk    (1UL /*<< ITMIRR_ATREADYM_Pos*/) /*!< ITM IRR:
ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */

#define ITM_IMCR_INTEGRATION_Pos 0                               /*!< ITM IMCR: INTEGRATION
Position */
#define ITM_IMCR_INTEGRATION_Msk (1UL /*<< ITM_IMCR_INTEGRATION_Pos*/) /*!< ITM
IMCR: INTEGRATION Mask */

```

```

/* ITM Lock Status Register Definitions */

#define ITM_LSR_ByteAcc_Pos      2                  /*!< ITM LSR: ByteAcc Position */
#define ITM_LSR_ByteAcc_Msk      (1UL << ITM_LSR_ByteAcc_Pos)      /*!< ITM LSR: ByteAcc Mask */

#define ITM_LSR_Access_Pos       1                  /*!< ITM LSR: Access Position */
#define ITM_LSR_Access_Msk      (1UL << ITM_LSR_Access_Pos)      /*!< ITM LSR: Access Mask */

#define ITM_LSR_Present_Pos     0                  /*!< ITM LSR: Present Position */
#define ITM_LSR_Present_Msk     (1UL /*<< ITM_LSR_Present_Pos*/)      /*!< ITM LSR: Present Mask */

/*}@*/ /* end of group CMSIS_ITM */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
 \brief Type definitions for the Data Watchpoint and Trace (DWT)

 @{
 */

```

```
/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).
```

```
*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) Control Register */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle Count Register */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI Count Register */
}
```

```

__IO uint32_t EXCCNT;           /*!< Offset: 0x00C (R/W) Exception Overhead Count Register */
__IO uint32_t SLEEP_CNT;        /*!< Offset: 0x010 (R/W) Sleep Count Register */
__IO uint32_t LSUCNT;           /*!< Offset: 0x014 (R/W) LSU Count Register */
__IO uint32_t FOLDCNT;          /*!< Offset: 0x018 (R/W) Folded-instruction Count Register */
__I uint32_t PCSR;              /*!< Offset: 0x01C (R) Program Counter Sample Register */
__IO uint32_t COMPO;             /*!< Offset: 0x020 (R/W) Comparator Register 0 */
__IO uint32_t MASK0;             /*!< Offset: 0x024 (R/W) Mask Register 0 */
__IO uint32_t FUNCTION0;         /*!< Offset: 0x028 (R/W) Function Register 0 */
                               uint32_t RESERVED0[1];
__IO uint32_t COMP1;             /*!< Offset: 0x030 (R/W) Comparator Register 1 */
__IO uint32_t MASK1;             /*!< Offset: 0x034 (R/W) Mask Register 1 */
__IO uint32_t FUNCTION1;         /*!< Offset: 0x038 (R/W) Function Register 1 */
                               uint32_t RESERVED1[1];
__IO uint32_t COMP2;             /*!< Offset: 0x040 (R/W) Comparator Register 2 */
__IO uint32_t MASK2;             /*!< Offset: 0x044 (R/W) Mask Register 2 */
__IO uint32_t FUNCTION2;         /*!< Offset: 0x048 (R/W) Function Register 2 */
                               uint32_t RESERVED2[1];
__IO uint32_t COMP3;             /*!< Offset: 0x050 (R/W) Comparator Register 3 */
__IO uint32_t MASK3;             /*!< Offset: 0x054 (R/W) Mask Register 3 */
__IO uint32_t FUNCTION3;         /*!< Offset: 0x058 (R/W) Function Register 3 */

} DWT_Type;

```

```

/* DWT Control Register Definitions */

#define DWT_CTRL_NUMCOMP_Pos      28           /*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk       (0xFUL << DWT_CTRL_NUMCOMP_Pos)    /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos     27           /*!< DWT CTRL: NOTRCPKT Position */

```

#define DWT_CTRL_NOTRCPKT_Msk	(0x1UL << DWT_CTRL_NOTRCPKT_Pos)	/*!< DWT
CTRL: NOTRCPKT Mask */		
#define DWT_CTRL_NOEXTTRIG_Pos	26	/*!< DWT CTRL: NOEXTTRIG
Position */		
#define DWT_CTRL_NOEXTTRIG_Msk	(0x1UL << DWT_CTRL_NOEXTTRIG_Pos)	/*!< DWT
CTRL: NOEXTTRIG Mask */		
#define DWT_CTRL_NOCYCCNT_Pos	25	/*!< DWT CTRL: NOCYCCNT
Position */		
#define DWT_CTRL_NOCYCCNT_Msk	(0x1UL << DWT_CTRL_NOCYCCNT_Pos)	/*!< DWT
CTRL: NOCYCCNT Mask */		
#define DWT_CTRL_NOPRFCNT_Pos	24	/*!< DWT CTRL: NOPRFCNT
Position */		
#define DWT_CTRL_NOPRFCNT_Msk	(0x1UL << DWT_CTRL_NOPRFCNT_Pos)	/*!< DWT
CTRL: NOPRFCNT Mask */		
#define DWT_CTRL_CYCEVTENA_Pos	22	/*!< DWT CTRL: CYCEVTENA
Position */		
#define DWT_CTRL_CYCEVTENA_Msk	(0x1UL << DWT_CTRL_CYCEVTENA_Pos)	/*!< DWT
CTRL: CYCEVTENA Mask */		
#define DWT_CTRL_FOLDEVTENA_Pos	21	/*!< DWT CTRL: FOLDEVTENA
Position */		
#define DWT_CTRL_FOLDEVTENA_Msk	(0x1UL << DWT_CTRL_FOLDEVTENA_Pos)	/*!< DWT
CTRL: FOLDEVTENA Mask */		
#define DWT_CTRL_LSUEVTENA_Pos	20	/*!< DWT CTRL: LSUEVTENA
Position */		
#define DWT_CTRL_LSUEVTENA_Msk	(0x1UL << DWT_CTRL_LSUEVTENA_Pos)	/*!< DWT
CTRL: LSUEVTENA Mask */		

#define DWT_CTRL_SLEEPEVTENA_Pos	19	/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk	(0x1UL << DWT_CTRL_SLEEPEVTENA_Pos)	/*!< DWT CTRL: SLEEPEVTENA Mask */
#define DWT_CTRL_EXCEVTENA_Pos	18	/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk	(0x1UL << DWT_CTRL_EXCEVTENA_Pos)	/*!< DWT CTRL: EXCEVTENA Mask */
#define DWT_CTRL_CPIEVTEA_Pos	17	/*!< DWT CTRL: CPIEVTEA Position */
#define DWT_CTRL_CPIEVTEA_Msk	(0x1UL << DWT_CTRL_CPIEVTEA_Pos)	/*!< DWT CTRL: CPIEVTEA Mask */
#define DWT_CTRL_EXCTRCENA_Pos	16	/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk	(0x1UL << DWT_CTRL_EXCTRCENA_Pos)	/*!< DWT CTRL: EXCTRCENA Mask */
#define DWT_CTRL_PCSAMPLENA_Pos	12	/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk	(0x1UL << DWT_CTRL_PCSAMPLENA_Pos)	/*!< DWT CTRL: PCSAMPLENA Mask */
#define DWT_CTRL_SYNCTAP_Pos	10	/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk	(0x3UL << DWT_CTRL_SYNCTAP_Pos)	/*!< DWT CTRL: SYNCTAP Mask */
#define DWT_CTRL_CYCTAP_Pos	9	/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk	(0x1UL << DWT_CTRL_CYCTAP_Pos)	/*!< DWT CTRL: CYCTAP Mask */

```

#define DWT_CTRL_POSTINIT_Pos      5          /*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk      (0xFUL << DWT_CTRL_POSTINIT_Pos)    /*!< DWT CTRL:
POSTINIT Mask */

#define DWT_CTRL_POSTPRESET_Pos    1          /*!< DWT CTRL: POSTPRESET
Position */
#define DWT_CTRL_POSTPRESET_Msk    (0xFUL << DWT_CTRL_POSTPRESET_Pos)    /*!< DWT
CTRL: POSTPRESET Mask */

#define DWT_CTRL_CYCCNTENA_Pos     0          /*!< DWT CTRL: CYCCNTENA
Position */
#define DWT_CTRL_CYCCNTENA_Msk    (0x1UL /*<< DWT_CTRL_CYCCNTENA_Pos*/) /*!< DWT
CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */

#define DWT_CPICNT_CPICNT_Pos     0          /*!< DWT CPICNT: CPICNT Position */
#define DWT_CPICNT_CPICNT_Msk     (0xFFUL /*<< DWT_CPICNT_CPICNT_Pos*/) /*!< DWT
CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */

#define DWT_EXCCNT_EXCCNT_Pos     0          /*!< DWT EXCCNT: EXCCNT
Position */
#define DWT_EXCCNT_EXCCNT_Msk     (0xFFUL /*<< DWT_EXCCNT_EXCCNT_Pos*/) /*!< DWT
EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */

#define DWT_SLEEPcnt_SLEEPcnt_Pos 0          /*!< DWT SLEEPcnt: SLEEPcnt
Position */
#define DWT_SLEEPcnt_SLEEPcnt_Msk (0xFFUL /*<< DWT_SLEEPcnt_SLEEPcnt_Pos*/) /*!<
DWT SLEEPcnt: SLEEPcnt Mask */

```

```

/* DWT LSU Count Register Definitions */

#define DWT_LSUCNT_LSUCNT_Pos      0           /*!< DWT LSUCNT: LSUCNT Position
*/
#define DWT_LSUCNT_LSUCNT_Msk     (0xFFUL /*<< DWT_LSUCNT_LSUCNT_Pos*/) /*!< DWT
LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */

#define DWT_FOLDCNT_FOLDCNT_Pos   0           /*!< DWT FOLDCNT: FOLDCNT
Position */
#define DWT_FOLDCNT_FOLDCNT_Msk   (0xFFUL /*<< DWT_FOLDCNT_FOLDCNT_Pos*/) /*!<
DWT FOLDCNT: FOLDCNT Mask */

/* DWT Comparator Mask Register Definitions */

#define DWT_MASK_MASK_Pos         0           /*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk         (0x1FUL /*<< DWT_MASK_MASK_Pos*/) /*!< DWT
MASK: MASK Mask */

/* DWT Comparator Function Register Definitions */

#define DWT_FUNCTION_MATCHED_Pos   24          /*!< DWT FUNCTION:
MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk   (0x1UL << DWT_FUNCTION_MATCHED_Pos) /*!<
DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16          /*!< DWT FUNCTION:
DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL << DWT_FUNCTION_DATAVADDR1_Pos) /*!<
DWT FUNCTION: DATAVADDR1 Mask */

#define DWT_FUNCTION_DATAVADDR0_Pos 12          /*!< DWT FUNCTION:
DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk (0xFUL << DWT_FUNCTION_DATAVADDR0_Pos) /*!<
DWT FUNCTION: DATAVADDR0 Mask */

```

```

#define DWT_FUNCTION_DATAVSIZE_Pos      10           /*!< DWT FUNCTION:  

DATAVSIZE Position */  
  

#define DWT_FUNCTION_DATAVSIZE_Msk     (0x3UL << DWT_FUNCTION_DATAVSIZE_Pos)  /*!<  

DWT FUNCTION: DATAVSIZE Mask */  
  

#define DWT_FUNCTION_LNK1ENA_Pos       9            /*!< DWT FUNCTION: LNK1ENA  

Position */  
  

#define DWT_FUNCTION_LNK1ENA_Msk      (0x1UL << DWT_FUNCTION_LNK1ENA_Pos)  /*!<  

DWT FUNCTION: LNK1ENA Mask */  
  

#define DWT_FUNCTION_DATAVMATCH_Pos    8            /*!< DWT FUNCTION:  

DATAVMATCH Position */  
  

#define DWT_FUNCTION_DATAVMATCH_Msk   (0x1UL << DWT_FUNCTION_DATAVMATCH_Pos)  
/*!< DWT FUNCTION: DATAVMATCH Mask */  
  

#define DWT_FUNCTION_CYCMATCH_Pos     7            /*!< DWT FUNCTION:  

CYCMATCH Position */  
  

#define DWT_FUNCTION_CYCMATCH_Msk    (0x1UL << DWT_FUNCTION_CYCMATCH_Pos)  /*!<  

DWT FUNCTION: CYCMATCH Mask */  
  

#define DWT_FUNCTION_EMITRANGE_Pos    5            /*!< DWT FUNCTION:  

EMITRANGE Position */  
  

#define DWT_FUNCTION_EMITRANGE_Msk   (0x1UL << DWT_FUNCTION_EMITRANGE_Pos)  /*!<  

DWT FUNCTION: EMITRANGE Mask */  
  

#define DWT_FUNCTION_FUNCTION_Pos     0            /*!< DWT FUNCTION:  

FUNCTION Position */  
  

#define DWT_FUNCTION_FUNCTION_Msk    (0xFUL /*<< DWT_FUNCTION_FUNCTION_Pos*/) /*!<  

DWT FUNCTION: FUNCTION Mask */  
  

/*@}*/ /* end of group CMSIS_DWT */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_TPI Trace Port Interface (TPI)
 \brief Type definitions for the Trace Port Interface (TPI)

 @{

 */

/** \brief Structure type to access the Trace Port Interface Register (TPI).
 */

typedef struct
{
    __IO uint32_t SSPSR;           /*!< Offset: 0x000 (R/ ) Supported Parallel Port Size Register */
    __IO uint32_t CSPSR;          /*!< Offset: 0x004 (R/W) Current Parallel Port Size Register */
    uint32_t RESERVED0[2];
    __IO uint32_t ACPR;           /*!< Offset: 0x010 (R/W) Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];
    __IO uint32_t SPPR;           /*!< Offset: 0x0F0 (R/W) Selected Pin Protocol Register */
    uint32_t RESERVED2[131];
    __I uint32_t FFSR;            /*!< Offset: 0x300 (R/ ) Formatter and Flush Status Register */
    __IO uint32_t FFCR;           /*!< Offset: 0x304 (R/W) Formatter and Flush Control Register */
    __I uint32_t FSCR;            /*!< Offset: 0x308 (R/ ) Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];
    __I uint32_t TRIGGER;         /*!< Offset: 0xEE8 (R/ ) TRIGGER */
    __I uint32_t FIFO0;           /*!< Offset: 0xEEC (R/ ) Integration ETM Data */
    __I uint32_t ITATBCTR2;       /*!< Offset: 0xEF0 (R/ ) ITATBCTR2 */
    uint32_t RESERVED4[1];
    __I uint32_t ITATBCTR0;       /*!< Offset: 0xEF8 (R/ ) ITATBCTR0 */
    __I uint32_t FIFO1;           /*!< Offset: 0xEFC (R/ ) Integration ITM Data */
    __IO uint32_t ITCTRL;          /*!< Offset: 0xF00 (R/W) Integration Mode Control */
    uint32_t RESERVED5[39];
}

```

```

__IO uint32_t CLAIMSET;           /*!< Offset: 0xFA0 (R/W) Claim tag set */
__IO uint32_t CLAIMCLR;          /*!< Offset: 0xFA4 (R/W) Claim tag clear */
uint32_t RESERVED7[8];

__I uint32_t DEVID;             /*!< Offset: 0xFC8 (R/ ) TPIU_DEVID */
__I uint32_t DEVTYPE;           /*!< Offset: 0xFCC (R/ ) TPIU_DEVTYPE */

} TPI_Type;

/* TPI Asynchronous Clock Prescaler Register Definitions */

#define TPI_ACPR_PRESCALER_Pos      0           /*!< TPI ACPR: PRESCALER Position */
#define TPI_ACPR_PRESCALER_Msk      (0xFFFFUL /*<< TPI_ACPR_PRESCALER_Pos*/) /*!< TPI
ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */

#define TPI_SPPR_TXMODE_Pos        0           /*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk        (0x3UL /*<< TPI_SPPR_TXMODE_Pos*/) /*!< TPI SPPR:
TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */

#define TPI_FFSR_FtNonStop_Pos     3           /*!< TPI FFSR: FtNonStop Position */
#define TPI_FFSR_FtNonStop_Msk     (0x1UL << TPI_FFSR_FtNonStop_Pos) /*!< TPI FFSR:
FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos     2           /*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk     (0x1UL << TPI_FFSR_TCPresent_Pos) /*!< TPI FFSR:
TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos     1           /*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk     (0x1UL << TPI_FFSR_FtStopped_Pos) /*!< TPI FFSR:
FtStopped Mask */

```

```

#define TPI_FFSR_FInProg_Pos      0           /*!< TPI FFSR: FInProg Position */
#define TPI_FFSR_FInProg_Msk     (0x1UL /*<< TPI_FFSR_FInProg_Pos*/)    /*!< TPI FFSR:
FInProg Mask */

/* TPI Formatter and Flush Control Register Definitions */

#define TPI_FFCR_TrigIn_Pos      8           /*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk     (0x1UL << TPI_FFCR_TrigIn_Pos)    /*!< TPI FFCR:
TrigIn Mask */

#define TPI_FFCR_EnFCont_Pos      1           /*!< TPI FFCR: EnFCont Position */
#define TPI_FFCR_EnFCont_Msk     (0x1UL << TPI_FFCR_EnFCont_Pos)    /*!< TPI FFCR:
EnFCont Mask */

/* TPI TRIGGER Register Definitions */

#define TPI_TRIGGER_TRIGGER_Pos      0           /*!< TPI TRIGGER: TRIGGER Position */
#define TPI_TRIGGER_TRIGGER_Msk   (0x1UL /*<< TPI_TRIGGER_TRIGGER_Pos*/)    /*!< TPI
TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */

#define TPI_FIFO0_ITM_ATVALID_Pos      29          /*!< TPI FIFO0: ITM_ATVALID
Position */
#define TPI_FIFO0_ITM_ATVALID_Msk   (0x3UL << TPI_FIFO0_ITM_ATVALID_Pos)    /*!< TPI FIFO0:
ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos      27          /*!< TPI FIFO0: ITM_bytecount
Position */
#define TPI_FIFO0_ITM_bytecount_Msk   (0x3UL << TPI_FIFO0_ITM_bytecount_Pos)    /*!< TPI
FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos      26          /*!< TPI FIFO0: ETM_ATVALID
Position */

```

```

#define TPI_FIFO0_ETM_ATVALID_Msk      (0x3UL << TPI_FIFO0_ETM_ATVALID_Pos)    /*!< TPI
FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos    24                                /*!< TPI FIFO0: ETM_bytecount
Position */

#define TPI_FIFO0_ETM_bytecount_Msk    (0x3UL << TPI_FIFO0_ETM_bytecount_Pos)  /*!< TPI
FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos            16                                /*!< TPI FIFO0: ETM2 Position */

#define TPI_FIFO0_ETM2_Msk            (0xFFUL << TPI_FIFO0_ETM2_Pos)        /*!< TPI FIFO0: ETM2
Mask */

#define TPI_FIFO0_ETM1_Pos            8       /*!< TPI FIFO0: ETM1 Position */

#define TPI_FIFO0_ETM1_Msk            (0xFFUL << TPI_FIFO0_ETM1_Pos)        /*!< TPI FIFO0: ETM1
Mask */

#define TPI_FIFO0_ETM0_Pos            0       /*!< TPI FIFO0: ETM0 Position */

#define TPI_FIFO0_ETM0_Msk            (0xFFUL /*<< TPI_FIFO0_ETM0_Pos*/)    /*!< TPI FIFO0:
ETM0 Mask */

/* TPI ITATBCTR2 Register Definitions */

#define TPI_ITATBCTR2_ATREADY_Pos     0           /*!< TPI ITATBCTR2: ATREADY
Position */

#define TPI_ITATBCTR2_ATREADY_Msk    (0x1UL /*<< TPI_ITATBCTR2_ATREADY_Pos*/) /*!< TPI
ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */

#define TPI_FIFO1_ITM_ATVALID_Pos    29          /*!< TPI FIFO1: ITM_ATVALID
Position */

#define TPI_FIFO1_ITM_ATVALID_Msk    (0x3UL << TPI_FIFO1_ITM_ATVALID_Pos)  /*!< TPI FIFO1:
ITM_ATVALID Mask */

```

```

#define TPI_FIFO1_ITM_bytecount_Pos    27          /*!< TPI FIFO1: ITM_bytecount
Position */

#define TPI_FIFO1_ITM_bytecount_Msk   (0x3UL << TPI_FIFO1_ITM_bytecount_Pos) /*!< TPI
FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos    26          /*!< TPI FIFO1: ETM_ATVALID
Position */

#define TPI_FIFO1_ETM_ATVALID_Msk   (0x3UL << TPI_FIFO1_ETM_ATVALID_Pos) /*!< TPI
FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos    24          /*!< TPI FIFO1: ETM_bytecount
Position */

#define TPI_FIFO1_ETM_bytecount_Msk   (0x3UL << TPI_FIFO1_ETM_bytecount_Pos) /*!< TPI
FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos           16          /*!< TPI FIFO1: ITM2 Position */

#define TPI_FIFO1_ITM2_Msk           (0xFFUL << TPI_FIFO1_ITM2_Pos) /*!< TPI FIFO1: ITM2
Mask */

#define TPI_FIFO1_ITM1_Pos           8           /*!< TPI FIFO1: ITM1 Position */

#define TPI_FIFO1_ITM1_Msk           (0xFFUL << TPI_FIFO1_ITM1_Pos) /*!< TPI FIFO1: ITM1
Mask */

#define TPI_FIFO1_ITM0_Pos           0           /*!< TPI FIFO1: ITM0 Position */

#define TPI_FIFO1_ITM0_Msk           (0xFFUL /*<< TPI_FIFO1_ITM0_Pos*/) /*!< TPI FIFO1: ITM0
Mask */

/* TPI ITATBCTRO Register Definitions */

#define TPI_ITATBCTRO_ATREADY_Pos    0          /*!< TPI ITATBCTRO: ATREADY
Position */

#define TPI_ITATBCTRO_ATREADY_Msk   (0x1UL /*<< TPI_ITATBCTRO_ATREADY_Pos*/) /*!< TPI
ITATBCTRO: ATREADY Mask */

```

```

/* TPI Integration Mode Control Register Definitions */

#define TPI_ITCTRL_Mode_Pos      0           /*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk      (0x1UL /*<< TPI_ITCTRL_Mode_Pos*/)    /*!< TPI ITCTRL:
Mode Mask */

/* TPI DEVID Register Definitions */

#define TPI_DEVID_NRZVALID_Pos   11          /*!< TPI DEVID: NRZVALID Position */
#define TPI_DEVID_NRZVALID_Msk   (0x1UL << TPI_DEVID_NRZVALID_Pos)    /*!< TPI DEVID:
NRZVALID Mask */

#define TPI_DEVID_MANCVALID_Pos  10          /*!< TPI DEVID: MANCVALID
Position */
#define TPI_DEVID_MANCVALID_Msk  (0x1UL << TPI_DEVID_MANCVALID_Pos)    /*!< TPI
DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALID_Pos   9           /*!< TPI DEVID: PTINVALID Position */
#define TPI_DEVID_PTINVALID_Msk   (0x1UL << TPI_DEVID_PTINVALID_Pos)    /*!< TPI DEVID:
PTINVALID Mask */

#define TPI_DEVID_MinBufSz_Pos   6           /*!< TPI DEVID: MinBufSz Position */
#define TPI_DEVID_MinBufSz_Msk   (0x7UL << TPI_DEVID_MinBufSz_Pos)    /*!< TPI DEVID:
MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos   5           /*!< TPI DEVID: AsynClkIn Position */
#define TPI_DEVID_AsynClkIn_Msk   (0x1UL << TPI_DEVID_AsynClkIn_Pos)    /*!< TPI DEVID:
AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos 0           /*!< TPI DEVID: NrTraceInput
Position */
#define TPI_DEVID_NrTraceInput_Msk (0xFUL /*<< TPI_DEVID_NrTraceInput_Pos*/) /*!< TPI
DEVID: NrTraceInput Mask */

```

```

/* TPI DEVTYPE Register Definitions */

#define TPI_DEVTYPE_MajorType_Pos      4          /*!< TPI DEVTYPE: MajorType Position */
#define TPI_DEVTYPE_MajorType_Msk      (0xFUL << TPI_DEVTYPE_MajorType_Pos) /*!< TPI DEVTYPE: MajorType Mask */

#define TPI_DEVTYPE_SubType_Pos       0          /*!< TPI DEVTYPE: SubType Position */
#define TPI_DEVTYPE_SubType_Msk      (0xFUL /*<< TPI_DEVTYPE_SubType_Pos*/) /*!< TPI DEVTYPE: SubType Mask */

/*}@*/ /* end of group CMSIS_TPI */

```

```

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_MPUs Memory Protection Unit (MPU)
 \brief Type definitions for the Memory Protection Unit (MPU)

@{

*/

/** \brief Structure type to access the Memory Protection Unit (MPU).
 */

typedef struct
{
    __I uint32_t TYPE;           /*!< Offset: 0x000 (R/ ) MPU Type Register */
    __IO uint32_t CTRL;          /*!< Offset: 0x004 (R/W) MPU Control Register */
    __IO uint32_t RNR;           /*!< Offset: 0x008 (R/W) MPU Region RNRber Register */
    __IO uint32_t RBAR;          /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
}
```

```

    __IO uint32_t RASR;           /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register
*/
    __IO uint32_t RBAR_A1;        /*!< Offset: 0x014 (R/W) MPU Alias 1 Region Base Address Register
*/
    __IO uint32_t RASR_A1;        /*!< Offset: 0x018 (R/W) MPU Alias 1 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A2;        /*!< Offset: 0x01C (R/W) MPU Alias 2 Region Base Address Register
*/
    __IO uint32_t RASR_A2;        /*!< Offset: 0x020 (R/W) MPU Alias 2 Region Attribute and Size
Register */
    __IO uint32_t RBAR_A3;        /*!< Offset: 0x024 (R/W) MPU Alias 3 Region Base Address Register
*/
    __IO uint32_t RASR_A3;        /*!< Offset: 0x028 (R/W) MPU Alias 3 Region Attribute and Size
Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)      /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)      /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)      /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

```

```

#define MPU_CTRL_PRIVDEFENA_Pos      2           /*!< MPU CTRL: PRIVDEFENA
Position */

#define MPU_CTRL_PRIVDEFENA_Msk     (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos      1           /*!< MPU CTRL: HFNMIENA
Position */

#define MPU_CTRL_HFNMIENA_Msk     (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos        0           /*!< MPU CTRL: ENABLE Position */

#define MPU_CTRL_ENABLE_Msk       (1UL /*<< MPU_CTRL_ENABLE_Pos*/)    /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */

#define MPU_RNR_REGION_Pos        0           /*!< MPU RNR: REGION Position */

#define MPU_RNR_REGION_Msk       (0xFFUL /*<< MPU_RNR_REGION_Pos*/)    /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos         5           /*!< MPU RBAR: ADDR Position */

#define MPU_RBAR_ADDR_Msk       (0x7FFFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos        4           /*!< MPU RBAR: VALID Position */

#define MPU_RBAR_VALID_Msk       (1UL << MPU_RBAR_VALID_Pos)    /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos        0           /*!< MPU RBAR: REGION Position */

#define MPU_RBAR_REGION_Msk       (0xFUL /*<< MPU_RBAR_REGION_Pos*/)    /*!< MPU
RBAR: REGION Mask */

```

```

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos      16          /*!< MPU RASR: MPU Region
Attribute field Position */

#define MPU_RASR_ATTRS_Msk      (0xFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28          /*!< MPU RASR: ATTRS.XN Position
*/
#define MPU_RASR_XN_Msk         (1UL << MPU_RASR_XN_Pos)        /*!< MPU RASR:
ATTRS.XN Mask */

#define MPU_RASR_AP_Pos         24          /*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk         (0x7UL << MPU_RASR_AP_Pos)       /*!< MPU RASR:
ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos        19          /*!< MPU RASR: ATTRS.TEX Position
*/
#define MPU_RASR_TEX_Msk        (0x7UL << MPU_RASR_TEX_Pos)       /*!< MPU RASR:
ATTRS.TEX Mask */

#define MPU_RASR_S_Pos           18          /*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk          (1UL << MPU_RASR_S_Pos)        /*!< MPU RASR: ATTRS.S
Mask */

#define MPU_RASR_C_Pos           17          /*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk          (1UL << MPU_RASR_C_Pos)        /*!< MPU RASR:
ATTRS.C Mask */

#define MPU_RASR_B_Pos           16          /*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk          (1UL << MPU_RASR_B_Pos)        /*!< MPU RASR:
ATTRS.B Mask */

```

```

#define MPU_RASR_SRD_Pos           8          /*!< MPU RASR: Sub-Region Disable
Position */

#define MPU_RASR_SRD_Msk          (0xFFUL << MPU_RASR_SRD_Pos)      /*!< MPU RASR:
Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos          1          /*!< MPU RASR: Region Size Field
Position */

#define MPU_RASR_SIZE_Msk          (0x1FUL << MPU_RASR_SIZE_Pos)      /*!< MPU RASR:
Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos         0          /*!< MPU RASR: Region enable bit
Position */

#define MPU_RASR_ENABLE_Msk         (1UL /*<< MPU_RASR_ENABLE_Pos*/)    /*!< MPU
RASR: Region enable bit Disable Mask */

/*@} end of group CMSIS_MPUS
#endif

```

```

#if (__FPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
\defgroup CMSIS_FPU Floating Point Unit (FPU)
\brief Type definitions for the Floating Point Unit (FPU)
@{
*/

```

```

/** \brief Structure type to access the Floating Point Unit (FPU).
*/
typedef struct
{

```

```

    uint32_t RESERVED0[1];

    __IO uint32_t FPCCR;           /*!< Offset: 0x004 (R/W) Floating-Point Context Control Register */
    */

    __IO uint32_t FPCAR;          /*!< Offset: 0x008 (R/W) Floating-Point Context Address Register */
    */

    __IO uint32_t FPDSCR;         /*!< Offset: 0x00C (R/W) Floating-Point Default Status Control
Register      */

    __I uint32_t MVFR0;          /*!< Offset: 0x010 (R/ ) Media and FP Feature Register 0      */
    __I uint32_t MVFR1;          /*!< Offset: 0x014 (R/ ) Media and FP Feature Register 1      */

} FPU_Type;

/* Floating-Point Context Control Register */

#define FPU_FPCCR_AS PEN_Pos      31           /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk       (1UL << FPU_FPCCR_AS PEN_Pos)      /*!< FPCCR: ASPEN
bit Mask */

#define FPU_FPCCR_LSP EN_Pos      30           /*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSP EN_Msk       (1UL << FPU_FPCCR_LSP EN_Pos)      /*!< FPCCR: LSPEN
bit Mask */

#define FPU_FPCCR_MON RDY_Pos     8            /*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MON RDY_Msk     (1UL << FPU_FPCCR_MON RDY_Pos)      /*!< FPCCR:
MONRDY bit Mask */

#define FPU_FPCCR_BFR DY_Pos      6            /*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFR DY_Msk       (1UL << FPU_FPCCR_BFR DY_Pos)      /*!< FPCCR: BFRDY
bit Mask */

#define FPU_FPCCR_MMR DY_Pos      5            /*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMR DY_Msk       (1UL << FPU_FPCCR_MMR DY_Pos)      /*!< FPCCR:
MMRDY bit Mask */

```

```

#define FPU_FPCCR_HFRDY_Pos      4          /*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk      (1UL << FPU_FPCCR_HFRDY_Pos)    /*!< FPCCR:
HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos     3          /*!< FPCCR: processor mode bit
Position */
#define FPU_FPCCR_THREAD_Msk     (1UL << FPU_FPCCR_THREAD_Pos)    /*!< FPCCR:
processor mode active bit Mask */

#define FPU_FPCCR_USER_Pos       1          /*!< FPCCR: privilege level bit
Position */
#define FPU_FPCCR_USER_Msk       (1UL << FPU_FPCCR_USER_Pos)    /*!< FPCCR:
privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos     0          /*!< FPCCR: Lazy state preservation
active bit Position */
#define FPU_FPCCR_LSPACT_Msk     (1UL /*<< FPU_FPCCR_LSPACT_Pos*/) /*!< FPCCR: Lazy
state preservation active bit Mask */

/* Floating-Point Context Address Register */

#define FPU_FPCAR_ADDRESS_Pos    3          /*!< FPCAR: ADDRESS bit Position */
#define FPU_FPCAR_ADDRESS_Msk    (0x1FFFFFFFUL << FPU_FPCAR_ADDRESS_Pos) /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */

#define FPU_FPDSCR_AHP_Pos       26         /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk       (1UL << FPU_FPDSCR_AHP_Pos)    /*!< FPDSCR: AHP
bit Mask */

#define FPU_FPDSCR_DN_Pos        25         /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk        (1UL << FPU_FPDSCR_DN_Pos)    /*!< FPDSCR: DN bit
Mask */

```

```

#define FPU_FPDSCR_FZ_Pos      24          /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk      (1UL << FPU_FPDSCR_FZ_Pos)      /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos   22          /*!< FPDSCR: RMode bit Position */
#define FPU_FPDSCR_RMode_Msk   (3UL << FPU_FPDSCR_RMode_Pos)      /*!< FPDSCR:
RMode bit Mask */

/* Media and FP Feature Register 0 */

#define FPU_MVFRO_FP_rounding_modes_Pos 28          /*!< MVFRO: FP rounding
modes bits Position */
#define FPU_MVFRO_FP_rounding_modes_Msk (0xFUL << FPU_MVFRO_FP_rounding_modes_Pos)
/*!< MVFRO: FP rounding modes bits Mask */

#define FPU_MVFRO_Short_vectors_Pos 24          /*!< MVFRO: Short vectors bits
Position */
#define FPU_MVFRO_Short_vectors_Msk (0xFUL << FPU_MVFRO_Short_vectors_Pos)      /*!<
MVFRO: Short vectors bits Mask */

#define FPU_MVFRO_Square_root_Pos 20          /*!< MVFRO: Square root bits
Position */
#define FPU_MVFRO_Square_root_Msk (0xFUL << FPU_MVFRO_Square_root_Pos)      /*!<
MVFRO: Square root bits Mask */

#define FPU_MVFRO_Divide_Pos     16          /*!< MVFRO: Divide bits Position */
#define FPU_MVFRO_Divide_Msk    (0xFUL << FPU_MVFRO_Divide_Pos)      /*!< MVFRO:
Divide bits Mask */

#define FPU_MVFRO_FP_excep_trapping_Pos 12          /*!< MVFRO: FP exception
trapping bits Position */

```

```

#define FPU_MVFR0_FP_excep_trapping_Msk  (0xFUL << FPU_MVFR0_FP_excep_trapping_Pos) /*!<
MVFR0: FP exception trapping bits Mask */

#define FPU_MVFR0_Double_precision_Pos    8                                /*!< MVFR0: Double-precision
bits Position */

#define FPU_MVFR0_Double_precision_Msk   (0xFUL << FPU_MVFR0_Double_precision_Pos) /*!<
MVFR0: Double-precision bits Mask */

#define FPU_MVFR0_Single_precision_Pos   4                                /*!< MVFR0: Single-precision
bits Position */

#define FPU_MVFR0_Single_precision_Msk  (0xFUL << FPU_MVFR0_Single_precision_Pos) /*!<
MVFR0: Single-precision bits Mask */

#define FPU_MVFR0_A SIMD_registers_Pos  0                                /*!< MVFR0: A SIMD registers
bits Position */

#define FPU_MVFR0_A SIMD_registers_Msk (0xFUL /*<< FPU_MVFR0_A SIMD_registers_Pos*/) /*!<
MVFR0: A SIMD registers bits Mask */

/* Media and FP Feature Register 1 */

#define FPU_MVFR1_FP_fused_MAC_Pos     28                               /*!< MVFR1: FP fused MAC
bits Position */

#define FPU_MVFR1_FP_fused_MAC_Msk   (0xFUL << FPU_MVFR1_FP_fused_MAC_Pos) /*!<
MVFR1: FP fused MAC bits Mask */

#define FPU_MVFR1_FP_HPFP_Pos        24                               /*!< MVFR1: FP HPFP bits
Position */

#define FPU_MVFR1_FP_HPFP_Msk       (0xFUL << FPU_MVFR1_FP_HPFP_Pos) /*!< MVFR1:
FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos     4                                /*!< MVFR1: D_NaN mode bits
Position */

#define FPU_MVFR1_D_NaN_mode_Msk   (0xFUL << FPU_MVFR1_D_NaN_mode_Pos) /*!<
MVFR1: D_NaN mode bits Mask */

```

```
#define FPU_MVFR1_FtZ_mode_Pos      0           /*!< MVFR1: FtZ mode bits
Position */

#define FPU_MVFR1_FtZ_mode_Msk      (0xFUL /*<< FPU_MVFR1_FtZ_mode_Pos*/) /*!<
MVFR1: FtZ mode bits Mask */

/*}@} end of group CMSIS_FPU */

#endif
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug    Core Debug Registers (CoreDebug)
\brief    Type definitions for the Core Debug Registers
@{
*/

/** \brief Structure type to access the Core Debug Register (CoreDebug).
*/
typedef struct
{
    __IO uint32_t DHCSR;          /*!< Offset: 0x000 (R/W) Debug Halting Control and Status Register */
    __O  uint32_t DCRSR;         /*!< Offset: 0x004 ( /W) Debug Core Register Selector Register */
    __IO uint32_t DCRDR;         /*!< Offset: 0x008 (R/W) Debug Core Register Data Register */
    __IO uint32_t DEMCR;         /*!< Offset: 0x00C (R/W) Debug Exception and Monitor Control
Register */
} CoreDebug_Type;

/* Debug Halting Control and Status Register */
```

```
#define CoreDebug_DHCSR_DBGKEY_Pos    16          /*!< CoreDebug DHCSR:  
DBGKEY Position */  
  
#define CoreDebug_DHCSR_DBGKEY_Msk    (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos)  /*!<  
CoreDebug DHCSR: DBGKEY Mask */  
  
  
#define CoreDebug_DHCSR_S_RESET_ST_Pos 25          /*!< CoreDebug DHCSR:  
S_RESET_ST Position */  
  
#define CoreDebug_DHCSR_S_RESET_ST_Msk (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RESET_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Pos 24          /*!< CoreDebug DHCSR:  
S_RETIRE_ST Position */  
  
#define CoreDebug_DHCSR_S_RETIRE_ST_Msk (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RETIRE_ST Mask */  
  
  
#define CoreDebug_DHCSR_S_LOCKUP_Pos   19          /*!< CoreDebug DHCSR:  
S_LOCKUP Position */  
  
#define CoreDebug_DHCSR_S_LOCKUP_Msk  (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)  /*!<  
CoreDebug DHCSR: S_LOCKUP Mask */  
  
  
#define CoreDebug_DHCSR_S_SLEEP_Pos   18          /*!< CoreDebug DHCSR:  
S_SLEEP Position */  
  
#define CoreDebug_DHCSR_S_SLEEP_Msk  (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)  /*!<  
CoreDebug DHCSR: S_SLEEP Mask */  
  
  
#define CoreDebug_DHCSR_S_HALT_Pos    17          /*!< CoreDebug DHCSR:  
S_HALT Position */  
  
#define CoreDebug_DHCSR_S_HALT_Msk  (1UL << CoreDebug_DHCSR_S_HALT_Pos)  /*!<  
CoreDebug DHCSR: S_HALT Mask */  
  
  
#define CoreDebug_DHCSR_S_REGRDY_Pos  16          /*!< CoreDebug DHCSR:  
S_REGRDY Position */  
  
#define CoreDebug_DHCSR_S_REGRDY_Msk (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)  /*!<  
CoreDebug DHCSR: S_REGRDY Mask */
```

```
#define CoreDebug_DHCSR_C_SNAPSTALL_Pos 5          /*!< CoreDebug DHCSR:  
C_SNAPSTALL Position */  
  
#define CoreDebug_DHCSR_C_SNAPSTALL_Msk (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)  
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */  
  
  
#define CoreDebug_DHCSR_C_MASKINTS_Pos 3           /*!< CoreDebug DHCSR:  
C_MASKINTS Position */  
  
#define CoreDebug_DHCSR_C_MASKINTS_Msk (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)  
/*!< CoreDebug DHCSR: C_MASKINTS Mask */  
  
  
#define CoreDebug_DHCSR_C_STEP_Pos 2                /*!< CoreDebug DHCSR: C_STEP  
Position */  
  
#define CoreDebug_DHCSR_C_STEP_Msk (1UL << CoreDebug_DHCSR_C_STEP_Pos)      /*!<  
CoreDebug DHCSR: C_STEP Mask */  
  
  
#define CoreDebug_DHCSR_C_HALT_Pos 1                /*!< CoreDebug DHCSR:  
C_HALT Position */  
  
#define CoreDebug_DHCSR_C_HALT_Msk (1UL << CoreDebug_DHCSR_C_HALT_Pos)      /*!<  
CoreDebug DHCSR: C_HALT Mask */  
  
  
#define CoreDebug_DHCSR_C_DEBUGEN_Pos 0             /*!< CoreDebug DHCSR:  
C_DEBUGEN Position */  
  
#define CoreDebug_DHCSR_C_DEBUGEN_Msk (1UL /*<< CoreDebug_DHCSR_C_DEBUGEN_Pos*/) /*!<  
CoreDebug DHCSR: C_DEBUGEN Mask */  
  
  
/* Debug Core Register Selector Register */  
  
#define CoreDebug_DCRSR_REGWnR_Pos 16              /*!< CoreDebug DCRSR:  
REGWnR Position */  
  
#define CoreDebug_DCRSR_REGWnR_Msk (1UL << CoreDebug_DCRSR_REGWnR_Pos)      /*!<  
CoreDebug DCRSR: REGWnR Mask */
```

```

#define CoreDebug_DCRSR_REGSEL_Pos      0           /*!< CoreDebug DCRSR: REGSEL Position */
#define CoreDebug_DCRSR_REGSEL_Msk     (0x1FUL /*<< CoreDebug_DCRSR_REGSEL_Pos*/) /*!< CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos    24          /*!< CoreDebug DEMCR: TRCENA Position */
#define CoreDebug_DEMCR_TRCENA_Msk   (1UL << CoreDebug_DEMCR_TRCENA_Pos) /*!< CoreDebug DEMCR: TRCENA Mask */

#define CoreDebug_DEMCR_MON_REQ_Pos   19          /*!< CoreDebug DEMCR: MON_REQ Position */
#define CoreDebug_DEMCR_MON_REQ_Msk  (1UL << CoreDebug_DEMCR_MON_REQ_Pos) /*!< CoreDebug DEMCR: MON_REQ Mask */

#define CoreDebug_DEMCR_MON_STEP_Pos  18          /*!< CoreDebug DEMCR: MON_STEP Position */
#define CoreDebug_DEMCR_MON_STEP_Msk (1UL << CoreDebug_DEMCR_MON_STEP_Pos) /*!< CoreDebug DEMCR: MON_STEP Mask */

#define CoreDebug_DEMCR_MON_PEND_Pos  17          /*!< CoreDebug DEMCR: MON_PEND Position */
#define CoreDebug_DEMCR_MON_PEND_Msk (1UL << CoreDebug_DEMCR_MON_PEND_Pos) /*!< CoreDebug DEMCR: MON_PEND Mask */

#define CoreDebug_DEMCR_MON_EN_Pos    16          /*!< CoreDebug DEMCR: MON_EN Position */
#define CoreDebug_DEMCR_MON_EN_Msk  (1UL << CoreDebug_DEMCR_MON_EN_Pos) /*!< CoreDebug DEMCR: MON_EN Mask */

#define CoreDebug_DEMCR_VC_HARDERR_Pos 10         /*!< CoreDebug DEMCR: VC_HARDERR Position */

```

```
#define CoreDebug_DEMCR_VC_HARDERR_Msk  (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos) /*!< CoreDebug DEMCR: VC_HARDERR Mask */

#define CoreDebug_DEMCR_VC_INTERR_Pos    9                         /*!< CoreDebug DEMCR: VC_INTERR Position */

#define CoreDebug_DEMCR_VC_INTERR_Msk   (1UL << CoreDebug_DEMCR_VC_INTERR_Pos) /*!< CoreDebug DEMCR: VC_INTERR Mask */

#define CoreDebug_DEMCR_VC_BUSERR_Pos   8                         /*!< CoreDebug DEMCR: VC_BUSERR Position */

#define CoreDebug_DEMCR_VC_BUSERR_Msk  (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos) /*!< CoreDebug DEMCR: VC_BUSERR Mask */

#define CoreDebug_DEMCR_VC_STATERR_Pos  7                         /*!< CoreDebug DEMCR: VC_STATERR Position */

#define CoreDebug_DEMCR_VC_STATERR_Msk (1UL << CoreDebug_DEMCR_VC_STATERR_Pos) /*!< CoreDebug DEMCR: VC_STATERR Mask */

#define CoreDebug_DEMCR_VC_CHKERR_Pos   6                         /*!< CoreDebug DEMCR: VC_CHKERR Position */

#define CoreDebug_DEMCR_VC_CHKERR_Msk  (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos) /*!< CoreDebug DEMCR: VC_CHKERR Mask */

#define CoreDebug_DEMCR_VC_NOCPERR_Pos  5                         /*!< CoreDebug DEMCR: VC_NOCPERR Position */

#define CoreDebug_DEMCR_VC_NOCPERR_Msk (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos) /*!< CoreDebug DEMCR: VC_NOCPERR Mask */

#define CoreDebug_DEMCR_VC_MMERR_Pos   4                         /*!< CoreDebug DEMCR: VC_MMERR Position */

#define CoreDebug_DEMCR_VC_MMERR_Msk  (1UL << CoreDebug_DEMCR_VC_MMERR_Pos) /*!< CoreDebug DEMCR: VC_MMERR Mask */
```

```

#define CoreDebug_DEMCR_VC_CORERESET_Pos 0          /*!< CoreDebug DEMCR:  

VC_CORERESET Position */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Msk (1UL /*<<  

CoreDebug_DEMCR_VC_CORERESET_Pos*/ ) /*!< CoreDebug DEMCR: VC_CORERESET Mask */  
  

/*@} end of group CMSIS_CoreDebug */  
  

  
  

/** \ingroup CMSIS_core_register  

\defgroup CMSIS_core_base Core Definitions  

\brief Definitions for base addresses, unions, and structures.  

@{  

*/  
  

/* Memory mapping of Cortex-M4 Hardware */  
  

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */  

#define ITM_BASE      (0xE0000000UL)           /*!< ITM Base Address */  

#define DWT_BASE      (0xE0001000UL)           /*!< DWT Base Address */  

#define TPI_BASE      (0xE0040000UL)           /*!< TPI Base Address */  

#define CoreDebug_BASE (0xE000EDFOUL)           /*!< Core Debug Base Address */  

#define SysTick_BASE  (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */  

#define NVIC_BASE     (SCS_BASE + 0x0100UL)       /*!< NVIC Base Address */  

#define SCB_BASE      (SCS_BASE + 0x0D00UL)       /*!< System Control Block Base Address */  
  

#define SCnSCB        ((SCnSCB_Type *) SCS_BASE) /*!< System control Register not in SCB */  

#define SCB           ((SCB_Type *) SCB_BASE)      /*!< SCB configuration struct */  

#define SysTick       ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */  

#define NVIC          ((NVIC_Type *) NVIC_BASE)      /*!< NVIC configuration struct */  

#define ITM           ((ITM_Type *) ITM_BASE)        /*!< ITM configuration struct */

```

```

#define DWT      ((DWT_Type *) DWT_BASE) /*!< DWT configuration struct */
#define TPI      ((TPI_Type *) TPI_BASE) /*!< TPI configuration struct */
#define CoreDebug ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration
struct */

#if (__MPU_PRESENT == 1)

#define MPU_BASE   (SCS_BASE + 0x0D90UL) /*!< Memory Protection Unit */
#define MPU       ((MPU_Type *) MPU_BASE) /*!< Memory Protection Unit */

#endif

#if (__FPU_PRESENT == 1)

#define FPU_BASE   (SCS_BASE + 0x0F30UL) /*!< Floating Point Unit */
#define FPU       ((FPU_Type *) FPU_BASE) /*!< Floating Point Unit */

#endif

/*@}*/

```

```

*****  

*      Hardware Abstraction Layer  

Core Function Interface contains:  

- Core NVIC Functions  

- Core SysTick Functions  

- Core Debug Functions  

- Core Register Access Functions  

*****/  

/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference  

*/

```

```

/* ##### NVIC functions #####
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the NVIC.

@{
*/

```

/** \brief Set Priority Grouping

The function sets the priority grouping field using the required unlock sequence.

The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.

Only values from 0..7 are used.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

\param [in] PriorityGroup Priority grouping field.

```

*/
__STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
    uint32_t reg_value;
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL);      /* only values 0..7 are used
*/
```

```

    reg_value = SCB->AIRCR;                      /* read old register configuration */
    reg_value &= ~((uint32_t)(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk)); /* clear
bits to change */
```

```

reg_value = (reg_value           |
((uint32_t)0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
(PriorityGroupTmp << 8)           );      /* Insert write key and priority group */
SCB->AIRCR = reg_value;
}

```

/** \brief Get Priority Grouping

The function reads the priority grouping field from the NVIC Interrupt Controller.

```

\return Priority grouping field (SCB->AIRCR [10:8] PRIGROUP field).
*/
__STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)
{
    return ((uint32_t)((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos));
}

```

/** \brief Enable External Interrupt

The function enables a device-specific interrupt in the NVIC interrupt controller.

```

\param [in] IRQn External interrupt number. Value cannot be negative.
*/
__STATIC_INLINE void NVIC_EnableIRQ(int32_t IRQn)
{
    NVIC->ISER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &
0x1FUL));
}

```

```
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ICER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
0x1FUL));
```

```
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit
for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
```

```
{
```

```
    return((uint32_t)((NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0UL);  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

\param [in] IRQn Interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
```

```
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

\param [in] IRQn External interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
```

```
{
```

```
    NVIC->ICPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
```

```
}
```

```
/** \brief Get Active Interrupt
```

The function reads the active register in NVIC and returns the active bit.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not active.

\return 1 Interrupt status is active.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
```

```
{
```

```
    return((uint32_t)((NVIC->IABR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL)) != 0UL) ? 1UL : 0UL);
```

```
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

```
{
```

```
    if((int32_t)IRQn < 0) {
```

```

    SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &
(uint32_t)0xFFUL);

}

else {

    NVIC->IP[((uint32_t)(int32_t)IRQn)]      = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &
(uint32_t)0xFFUL);

}

}

```

/ \brief Get Interrupt Priority**

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

\param [in] IRQn Interrupt number.
\return Interrupt Priority. Value is aligned automatically to the implemented priority bits of the microcontroller.

*/

__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)

{

if((int32_t)IRQn < 0) {

 return(((uint32_t)SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL) >> (8 - __NVIC_PRIO_BITS));

}

else {

 return(((uint32_t)NVIC->IP[((uint32_t)(int32_t)IRQn)] >> (8 - __NVIC_PRIO_BITS)));

```
}
```

```
/** \brief Encode Priority
```

The function encodes the priority for an interrupt with the given priority group, preemptive priority value, and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

```
\param [in] PriorityGroup Used priority group.  
\param [in] PreemptPriority Preemptive priority value (starting from 0).  
\param [in] SubPriority Subpriority value (starting from 0).  
\return Encoded priority. Value can be used in the function \ref NVIC_SetPriority().  
*/  
__STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority,  
uint32_t SubPriority)  
{  
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */  
    uint32_t PreemptPriorityBits;  
    uint32_t SubPriorityBits;  
  

```

```

        ((SubPriority & (uint32_t)((1UL << (SubPriorityBits )) - 1UL)))
    );
}

}

```

*/** \brief Decode Priority*

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.

In case of a conflict between priority grouping and available priority bits (`__NVIC_PRIO_BITS`) the smallest possible priority group is set.

`\param [in] Priority` Priority value, which can be retrieved with the function `\ref NVIC_GetPriority()`.

`\param [in] PriorityGroup` Used priority group.

`\param [out] pPreemptPriority` Preemptive priority value (starting from 0).

`\param [out] pSubPriority` Subpriority value (starting from 0).

`*/`

`__STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)`

`{`

`uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */`

`uint32_t PreemptPriorityBits;`

`uint32_t SubPriorityBits;`

`PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
 (uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);`

`SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
 (uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));`

```
*pPreemptPriority = (Priority >> SubPriorityBits) & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL);
*pSubPriority   = (Priority           ) & (uint32_t)((1UL << (SubPriorityBits )) - 1UL);
}
```

```
/** \brief System Reset
```

The function initiates a system reset request to reset the MCU.

```
*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                /* Ensure all outstanding memory accesses included
                                              buffered write are completed before reset */

    SCB->AIRCR = (uint32_t)((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                           (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
                           SCB_AIRCR_SYSRESETREQ_Msk);      /* Keep priority group unchanged */

    __DSB();                                /* Ensure completion of memory access */

    while(1) { __NOP(); }                    /* wait until reset */
}
```

```
/*@} end of CMSIS_Core_NVICFunctions */
```

```
/* ##### SysTick function
#####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.
```

```

@{

*/



#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0

/** \brief System Tick Configuration

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.
Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.
\return 1 Function failed.

\note When the variable <b>__Vendor_SysTickConfig</b> is set to 1, then the
function <b>SysTick_Config</b> is not included. In this case, the file <b><i>device</i>.h</b>
must contain a vendor-specific implementation of this function.

*/
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) { return (1UL); } /* Reload value impossible */

    SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
    SysTick->VAL = 0UL; /* Load the SysTick Counter Value */
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |

```

```
    SysTick_CTRL_TICKINT_Msk |  
    SysTick_CTRL_ENABLE_Msk;           /* Enable SysTick IRQ and SysTick Timer */  
    return (OUL);                      /* Function successful */  
}
```

```
#endif
```

```
/*}@} end of CMSIS_Core_SysTickFunctions */
```

```
/* ##### Debug In/Output function  
##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface  
 \defgroup CMSIS_core_DebugFunctions ITM Functions  
 \brief Functions that access the ITM debug interface.  
 @{  
 */
```

```
extern volatile int32_t ITM_RxBuffer;          /*!< External variable to receive characters.  
 */
```

```
#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is  
 ready for next character. */
```

```
/** \brief ITM Send Character
```

The function transmits a character via the ITM channel 0, and
\li Just returns when no debugger is connected that has booked the output.

\li Is blocking when a debugger is connected, but the previous character sent has not been transmitted.

\param [in] ch Character to transmit.

\returns Character to transmit.

*/

```
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)
{
    if (((ITM->TCR & ITM_TCR_ITMENA_Msk) != OUL) && /* ITM enabled */
        ((ITM->TER & 1UL ) != OUL) ) /* ITM Port #0 enabled */
    {
        while (ITM->PORT[0].u32 == OUL) { __NOP(); }

        ITM->PORT[0].u8 = (uint8_t)ch;
    }
    return (ch);
}
```

/** \brief ITM Receive Character

The function inputs a character via the external variable \ref ITM_RxBuffer.

\return Received character.

\return -1 No character pending.

*/

```
__STATIC_INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1; /* no character available */
```

```

if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
    ch = ITM_RxBuffer;
    ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
}

return (ch);
}

```

/** \brief ITM Check Character

The function checks whether a character is pending for reading in the variable \ref ITM_RxBuffer.

```

\return 0 No character available.

\return 1 Character available.

*/
__STATIC_INLINE int32_t ITM_CheckChar (void) {

if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {
    return (0); /* no character available */
} else {
    return (1); /* character available */
}

/*@} end of CMSIS_core_DebugFunctions */

```

```
#ifdef __cplusplus
}

#endif /* __CORE_CM4_H_DEPENDANT */

#endif /* __CMSIS_GENERIC */

/********************* @file core_cmFunc.h
 * @brief CMSIS Cortex-M Core Function Access Header File
 * @version V4.10
 * @date 18. March 2015
 *
 * @note
 *
 ****/
/* Copyright (c) 2009 - 2015 ARM LIMITED
```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMFUNC_H
#define __CORE_CMFUNC_H

/* ##### Core Function Access #####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_RegAccFunctions CMSIS Core Register Access Functions
 @{
 */

#if defined (__CC_ARM) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */
```

```
#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif
```

```
/* intrinsic void __enable_irq(); */
/* intrinsic void __disable_irq(); */
```

```
/** \brief Get Control Register
```

This function returns the content of the Control Register.

```
\return Control Register value
*/
__STATIC_INLINE uint32_t __get_CONTROL(void)
{
    register uint32_t __regControl      __ASM("control");
    return(__regControl);
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in] control Control Register value to set
*/
__STATIC_INLINE void __set_CONTROL(uint32_t control)
{
    register uint32_t __regControl      __ASM("control");
```

```
__regControl = control;  
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return IPSR Register value  
*/  
STATIC_INLINE uint32_t __get_IPSR(void)  
{  
    register uint32_t __regIPSR      __ASM("ipsr");  
    return(__regIPSR);  
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return APSR Register value  
*/  
STATIC_INLINE uint32_t __get_APSR(void)  
{  
    register uint32_t __regAPSR      __ASM("apsr");  
    return(__regAPSR);  
}
```

```
/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
\return xPSR Register value
*/
__STATIC_INLINE uint32_t __get_xPSR(void)
{
    register uint32_t __regXPSR      __ASM("xpsr");
    return(__regXPSR);
}
```

```
/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
\return PSP Register value
*/
__STATIC_INLINE uint32_t __get_PSP(void)
{
    register uint32_t __regProcessStackPointer __ASM("psp");
    return(__regProcessStackPointer);
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in] topOfProcStack Process Stack Pointer value to set
*/
__STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)
{
    register uint32_t __regProcessStackPointer __ASM("psp");
    __regProcessStackPointer = topOfProcStack;
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value
*/
__STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t __regMainStackPointer __ASM("msp");
    return(__regMainStackPointer);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
```

```
*/  
  
__STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)  
{  
    register uint32_t __regMainStackPointer __ASM("msp");  
    __regMainStackPointer = topOfMainStack;  
}
```

/** \brief Get Priority Mask

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value  
*/  
  
__STATIC_INLINE uint32_t __get_PRIMASK(void)  
{  
    register uint32_t __regPriMask __ASM("primask");  
    return(__regPriMask);  
}
```

/** \brief Set Priority Mask

This function assigns the given value to the Priority Mask Register.

```
\param [in] priMask Priority Mask  
*/  
  
__STATIC_INLINE void __set_PRIMASK(uint32_t priMask)  
{
```

```
register uint32_t __regPriMask      __ASM("primask");  
__regPriMask = (priMask);  
}
```

```
#if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
#define __enable_fault_irq      __enable_fiq
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
#define __disable_fault_irq     __disable_fiq
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

\return Base Priority register value

```
*/
```

```
__STATIC_INLINE uint32_t __get_BASEPRI(void)
{
    register uint32_t __regBasePri      __ASM("basepri");
    return(__regBasePri);
}
```

```
/** \brief Set Base Priority
```

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set
*/
__STATIC_INLINE void __set_BASEPRI(uint32_t basePri)
{
    register uint32_t __regBasePri      __ASM("basepri");
    __regBasePri = (basePri & 0xff);
}
```

```
/** \brief Set Base Priority with condition
```

This function assigns the given value to the Base Priority register only if BASEPRI masking is disabled, or the new value increases the BASEPRI priority level.

```
\param [in] basePri Base Priority value to set
*/
__STATIC_INLINE void __set_BASEPRI_MAX(uint32_t basePri)
{
```

```
register uint32_t __regBasePriMax    __ASM("basepri_max");
__regBasePriMax = (basePri & 0xff);
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return      Fault Mask register value
*/
__STATIC_INLINE uint32_t __get_FAULTMASK(void)
{
    register uint32_t __regFaultMask    __ASM("faultmask");
    return(__regFaultMask);
}
```

```
/** \brief Set Fault Mask
```

This function assigns the given value to the Fault Mask register.

```
\param [in]  faultMask  Fault Mask value to set
*/
__STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    register uint32_t __regFaultMask    __ASM("faultmask");
    __regFaultMask = (faultMask & (uint32_t)1);
}
```

```
#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */
```

```
#if    (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)
```

```
/** \brief Get FPSCR
```

This function returns the current value of the Floating Point Status/Control register.

```
\return      Floating Point Status/Control register value
*/
__STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpscr      __ASM("fpscr");
    return(__regfpscr);
#else
    return(0);
#endif
}
```

```
/** \brief Set FPSCR
```

This function assigns the given value to the Floating Point Status/Control register.

```
\param [in]  fpscr Floating Point Status/Control value to set
*/
```

```

__STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    register uint32_t __regfpSCR      __ASM("fpSCR");
    __regfpSCR = (fpSCR);
#endif
}

#endif /* (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07) */

#ifndef __GNUC__
/*----- GNU Compiler -----*/
/* GNU gcc specific functions */

/** \brief Enable IRQ Interrupts
 */

```

This function enables IRQ interrupts by clearing the I-bit in the CPSR.

Can only be executed in Privileged modes.

```

__attribute__(( always_inline )) __STATIC_INLINE void __enable_irq(void)
{
    __ASM volatile ("cpsie i" : : : "memory");
}

```

```

/** \brief Disable IRQ Interrupts
 */

```

This function disables IRQ interrupts by setting the I-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_irq(void)  
{  
    __ASM volatile ("cpsid i" : : : "memory");  
}
```

```
/** \brief Get Control Register
```

This function returns the content of the Control Register.

```
\return      Control Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_CONTROL(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, control" : "=r" (result));  
    return(result);  
}
```

```
/** \brief Set Control Register
```

This function writes the given value to the Control Register.

```
\param [in]  control Control Register value to set  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_CONTROL(uint32_t control)
```

```
{  
    __ASM volatile ("MSR control, %0" :: "r" (control) : "memory");  
}
```

```
/** \brief Get IPSR Register
```

This function returns the content of the IPSR Register.

```
\return      IPSR Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_IPSR(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, ipsr" : "=r" (result) );  
    return(result);  
}
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return      APSR Register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_APSR(void)  
{  
    uint32_t result;
```

```
    __ASM volatile ("MRS %0, apsr" : "=r" (result) );
    return(result);
}
```

```
/** \brief Get xPSR Register
```

This function returns the content of the xPSR Register.

```
    \return      xPSR Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_xPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, xpsr" : "=r" (result) );
    return(result);
}
```

```
/** \brief Get Process Stack Pointer
```

This function returns the current value of the Process Stack Pointer (PSP).

```
    \return      PSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_PSP(void)
{
```

```
register uint32_t result;

__ASM volatile ("MRS %0, psp\n" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Process Stack Pointer
```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in] topOfProcStack Process Stack Pointer value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)
{
    __ASM volatile ("MSR psp, %0\n" :: "r" (topOfProcStack) : "sp");
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return MSP Register value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t result;
```

```
__ASM volatile ("MRS %0, msp\n" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in] topOfMainStack Main Stack Pointer value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)
{
    __ASM volatile ("MSR msp, %0\n" :: "r" (topOfMainStack) : "sp");
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return Priority Mask value
*/
__attribute__(( always_inline )) __STATIC_INLINE uint32_t __get_PRIMASK(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, primask" : "=r" (result) );
    return(result);
```

```
}
```

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in] priMask Priority Mask
```

```
*/
```

```
__attribute__(( always_inline )) __STATIC_INLINE void __set_PRIMASK(uint32_t priMask)
```

```
{
```

```
__ASM volatile ("MSR primask, %0" :: "r" (priMask) : "memory");
```

```
}
```

```
#if __CORTEX_M >= 0x03
```

```
/** \brief Enable FIQ
```

This function enables FIQ interrupts by clearing the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/
```

```
__attribute__(( always_inline )) __STATIC_INLINE void __enable_fault_irq(void)
```

```
{
```

```
__ASM volatile ("cpsie f" :: : "memory");
```

```
}
```

```
/** \brief Disable FIQ
```

This function disables FIQ interrupts by setting the F-bit in the CPSR.

Can only be executed in Privileged modes.

```
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE void __disable_fault_irq(void)  
{  
    __ASM volatile ("cpsid f" :: : "memory");  
}
```

```
/** \brief Get Base Priority
```

This function returns the current value of the Base Priority register.

```
\return      Base Priority register value  
*/  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_BASEPRI(void)  
{  
    uint32_t result;  
  
    __ASM volatile ("MRS %0, basepri" : "=r" (result));  
    return(result);  
}
```

```
/** \brief Set Base Priority
```

This function assigns the given value to the Base Priority register.

```
\param [in] basePri Base Priority value to set  
*/  
attribute__( always_inline ) __STATIC_INLINE void __set_BASEPRI(uint32_t value)  
{  
    __ASM volatile ("MSR basepri, %0" :: "r" (value) : "memory");  
}
```

```
/** \brief Set Base Priority with condition
```

This function assigns the given value to the Base Priority register only if BASEPRI masking is disabled, or the new value increases the BASEPRI priority level.

```
\param [in] basePri Base Priority value to set  
*/  
attribute__( always_inline ) __STATIC_INLINE void __set_BASEPRI_MAX(uint32_t value)  
{  
    __ASM volatile ("MSR basepri_max, %0" :: "r" (value) : "memory");  
}
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask register.

```
\return Fault Mask register value  
*/  
attribute__( always_inline ) __STATIC_INLINE uint32_t __get_FAULTMASK(void)  
{
```

```
uint32_t result;

__ASM volatile ("MRS %0, faultmask" : "=r" (result) );
return(result);
}
```

```
/** \brief Set Fault Mask
```

This function assigns the given value to the Fault Mask register.

```
\param [in] faultMask Fault Mask value to set
*/
__attribute__(( always_inline )) __STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    __ASM volatile ("MSR faultmask, %0" : : "r" (faultMask) : "memory");
}

#endif /* (__CORTEX_M >= 0x03) */
```

```
#if    (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)
```

```
/** \brief Get FPSCR
```

This function returns the current value of the Floating Point Status/Control register.

```
\return      Floating Point Status/Control register value
*/
```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    uint32_t result;

    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMRS %0, fpSCR" : "=r" (result));
    __ASM volatile ("");
    return(result);
#else
    return(0);
#endif
}

```

/** \brief Set FPSCR

This function assigns the given value to the Floating Point Status/Control register.

```

\param [in] fpSCR Floating Point Status/Control value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMSR fpSCR, %0" : : "r" (fpSCR) : "vfpcc");
    __ASM volatile ("");

```

```
#endif
}

#endif /* (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07) */

#ifndef __CMSIS_H
#define __CMSIS_H

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
 */
#endif /* __CMSIS_H */
```



```
#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>
```

```
#endif

/*@} end of CMSIS_Core_RegAccFunctions */

#endif /* __CORE_CMFUNC_H */

/*********************//**/

* @file core_cm4_simd.h
* @brief CMSIS Cortex-M4 SIMD Header File
* @version V3.30
* @date 17. February 2014
*
* @note
*
*****/
```

/* Copyright (c) 2009 - 2014 ARM LIMITED

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif

#ifndef __CORE_CM4 SIMD_H
#define __CORE_CM4 SIMD_H

#endif /* cplusplus

extern "C" {

#endif

/*********************  

*      Hardware Abstraction Layer  

********************/  


```

```
/* ##### Compiler specific Intrinsics ##### */
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
Access to dedicated SIMD instructions

@{

*/
#ifndef __CMSIS_SIMD_H
#define __CMSIS_SIMD_H

/*if defined ( __CC_ARM ) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#define __SADD8          __sadd8
#define __QADD8          __qadd8
#define __SHADD8         __shadd8
#define __UADD8          __uadd8
#define __UQADD8         __uqadd8
#define __UHADD8         __uhadd8
#define __SSUB8          __ssub8
#define __QSUB8          __qsub8
#define __SHSUB8         __shsub8
#define __USUB8          __usub8
#define __UQSUB8         __uqsub8
#define __UHSUB8         __uhsub8
#define __SADD16         __sadd16
#define __QADD16         __qadd16
#define __SHADD16        __shadd16
#define __UADD16         __uadd16
#define __UQADD16        __uqadd16
#define __UHADD16        __uhadd16
#define __SSUB16         __ssub16
#define __QSUB16         __qsub16

#endif /* __CMSIS_SIMD_H */
```

```
#define __SHSUB16          __shsub16
#define __USUB16           __usub16
#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX              __sasx
#define __QASX              __qasx
#define __SHASX             __shasx
#define __UASX              __uasx
#define __UQASX             __uqasx
#define __UHASX             __uhasx
#define __SSAX               __ssax
#define __QSAX               __qsax
#define __SHSAX              __shsax
#define __USAX               __usax
#define __UQSAX              __uqsax
#define __UHSAX              __uhsax
#define __USAD8              __usad8
#define __USADA8             __usada8
#define __SSAT16             __ssat16
#define __USAT16             __usat16
#define __UXTB16             __uxtb16
#define __UXTAB16            __uxtab16
#define __SXTB16             __sxtb16
#define __SXTAB16            __sxtab16
#define __SMUAD              __smuad
#define __SMUADX             __smuadx
#define __SMLAD              __smlad
#define __SMLADX             __smladx
#define __SMLALD             __smlald
```

```

#define __SMLALDX      __smlaldx
#define __SMUSD         __smusd
#define __SMUSDX        __smusdx
#define __SMLSD          __smlsd
#define __SMLSDX         __smlsdx
#define __SMLS LD        __smlsld
#define __SMLS LDX       __smlsldx
#define __SEL            __sel
#define __QADD           __qadd
#define __QSUB           __qsub

#define __PKHBT(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0x0000FFFFUL) | \
                                (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)    ( (((uint32_t)(ARG1))      ) & 0xFFFF0000UL) | \
                                (((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL) )

#define __SMMLA(ARG1,ARG2,ARG3)    ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                ((int64_t)(ARG3) << 32) ) >> 32))

# elif defined ( __GNUC__ ) /*----- GNU Compiler -----*/
/* GNU gcc specific functions */

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

```

```
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
}
```

```
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSUB16(uint32_t op1, uint32_t op2)
{
```

```
uint32_t result;

__ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;
```

```
__ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAX(uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UQSAX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UHSAX(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __USAD8(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}
```

```

}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __USADA8(uint32_t op1, uint32_t op2,
uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#define __SSAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

#define __USAT16(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __UXTB16(uint32_t op1)
{
    uint32_t result;

```

```
__ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));  
return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __UXTAB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTB16(uint32_t op1)  
{  
    uint32_t result;  
  
    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUAD (uint32_t op1, uint32_t op2)
```

```
{  
    uint32_t result;  
  
    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMUADX (uint32_t op1, uint32_t op2)  
{  
    uint32_t result;  
  
    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLAD (uint32_t op1, uint32_t op2,  
uint32_t op3)  
{  
    uint32_t result;  
  
    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));  
    return(result);  
}  
  
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLADX (uint32_t op1, uint32_t op2,  
uint32_t op3)  
{  
    uint32_t result;
```

```

__ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));
return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALD (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLALDX (uint32_t op1, uint32_t op2,
uint64_t acc)
{
union llreg_u{
    uint32_t w32[2];
    uint64_t w64;
}

```

```

} llr;
llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );
#else // Big endian
__ASM volatile ("smalldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );
#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMUSD (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMUSDX (uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
    return(result);
}

```

```

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD (uint32_t op1, uint32_t op2,
uint32_t op3)

{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));

    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX (uint32_t op1, uint32_t op2,
uint32_t op3)

{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r" (op2), "r" (op3));

    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLS LD (uint32_t op1, uint32_t op2,
uint64_t acc)

{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;
}

#ifndef __ARMEB__ // Little endian

```

```

__ASM volatile ("smllsd %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else      // Big endian

__ASM volatile ("smllsd %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint64_t __SMLSLDX (uint32_t op1, uint32_t op2,
uint64_t acc)

{
union llreg_u{

    uint32_t w32[2];

    uint64_t w64;

} llr;

llr.w64 = acc;

#endifdef __ARMEB__ // Little endian

__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r" (llr.w32[1]): "r" (op1), "r" (op2) , "0"
(llr.w32[0]), "1" (llr.w32[1]) );

#else      // Big endian

__ASM volatile ("smllsidx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r" (llr.w32[0]): "r" (op1), "r" (op2) , "0"
(llr.w32[1]), "1" (llr.w32[0]) );

#endif

return(llr.w64);
}

```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SEL( uint32_t op1, uint32_t op2 )
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QADD(uint32_t op1, uint32_t op2)
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
__attribute__( __always_inline ) __STATIC_INLINE uint32_t __QSUB(uint32_t op1, uint32_t op2)
{
```

```
    uint32_t result;
```

```
    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2));
```

```
    return(result);
```

```
}
```

```
#define __PKHBT(ARG1,ARG2,ARG3) \
```

```
{ \
```

```
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
```

```
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "l" (ARG3) ); \
```

```
    __RES; \
```

```
})
```

```
#define __PKHTB(ARG1,ARG2,ARG3) \
{ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2) ); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" (__ARG2), "I" (ARG3) ); \
    __RES; \
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMMLA (int32_t op1, int32_t op2, int32_t
op3)
{
    int32_t result;

    __ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" (op2), "r" (op3));
    return(result);
}

#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
```

```
#include <cmsis_ccs.h>

#if defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/* not yet supported */

#if defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD_intrinsics */

#ifndef __cplusplus
}

#endif /* __CORE_CM4 SIMD_H */
//********************************************************************//**
* @file core_cm7.h
* @brief CMSIS Cortex-M7 Core Peripheral Access Layer Header File
* @version V4.10
* @date 18. March 2015
*
* @note
```

*

******/

/* Copyright (c) 2009 - 2015 ARM LIMITED

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )

#pragma system_include /* treat file as system include file for MISRA check */

#endif
```

```
#ifndef __CORE_CM7_H_GENERIC

#define __CORE_CM7_H_GENERIC
```

```
#ifdef __cplusplus

extern "C" {

#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions

CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.

Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union type: '...'.

Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.

Function-like macros are used to allow more efficient code.
*/

```
*****  
*      CMSIS definitions  
*****/*
```

```

/** \ingroup Cortex_M7

@{

*/


/* CMSIS CM7 definitions */

#define __CM7_CMSIS_VERSION_MAIN (0x04)           /*!< [31:16] CMSIS HAL main version */
#define __CM7_CMSIS_VERSION_SUB (0x00)             /*!< [15:0] CMSIS HAL sub version */
#define __CM7_CMSIS_VERSION ((__CM7_CMSIS_VERSION_MAIN << 16) | \
                           __CM7_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */

#define __CORTEX_M      (0x07)          /*!< Cortex-M Core */

#if defined (__CC_ARM)

#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)

#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)

#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

```

```

#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

#define __packed

#define __ASM      __asm           /*!< asm keyword for COSMIC Compiler */
#define __INLINE   inline          /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */

#define __STATIC_INLINE static inline

#endif

```

/** __FPU_USED indicates whether an FPU is used or not.

For this, __FPU_PRESENT has to be checked prior to making use of FPU specific registers and functions.

```

*/
#if defined ( __CC_ARM )
#if defined __TARGET_FPU_VFP
#if (__FPU_PRESENT == 1)
#define __FPU_USED    1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"

```

```

#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif

#elif defined ( __GNUC__ )
#if defined ( __VFP_FP__ ) && !defined( __SOFTFP__ )
#if ( __FPU_PRESENT == 1 )
#define __FPU_USED 1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED 0
#endif
#else
#define __FPU_USED 0
#endif

#elif defined ( __ICCARM__ )
#if defined __ARMVFP__
#if ( __FPU_PRESENT == 1 )
#define __FPU_USED 1
#else
#warning "Compiler generates FPU instructions for a device without an FPU (check
__FPU_PRESENT)"
#define __FPU_USED 0
#endif
#else

```

```
#define __FPU_USED    0
#endif

#ifndef __TMS470__
#define __FPU_USED    0
#endif

#define __FPU_USED    0
#endif
```

```

#if ( __CSMC__ & 0x400)           // FPU present for parser

#if ( __FPU_PRESENT == 1)

#define __FPU_USED    1

#else

#error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"

#define __FPU_USED    0

#endif

#endif

#define __FPU_USED    0

#endif

#endif

#include <stdint.h>          /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access       */
#include <core_cmFunc.h>         /* Core Function Access          */
#include <core_cmSimd.h>         /* Compiler specific SIMD Intrinsics */

#endif /* __cplusplus

}

#endif /* __CORE_CM7_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM7_H_DEPENDANT

#define __CORE_CM7_H_DEPENDANT

#endif /* __cplusplus

```

```
extern "C" {

#endif

/* check device defines and use defaults */

#if defined __CHECK_DEVICE_DEFINES

#ifndef __CM7_REV
#define __CM7_REV      0x0000
#warning "__CM7_REV not defined in device header file; using default!"

#endif

#ifndef __FPU_PRESENT
#define __FPU_PRESENT    0
#warning "__FPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT    0
#warning "__MPU_PRESENT not defined in device header file; using default!"

#endif

#ifndef __ICACHE_PRESENT
#define __ICACHE_PRESENT 0
#warning "__ICACHE_PRESENT not defined in device header file; using default!"

#endif

#ifndef __DCACHE_PRESENT
#define __DCACHE_PRESENT 0
#warning "__DCACHE_PRESENT not defined in device header file; using default!"

#endif
```

```

#ifndef __DTCM_PRESENT
#define __DTCM_PRESENT      0
#warning "__DTCM_PRESENT      not defined in device header file; using default!"
#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      3
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig  0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif

#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
 \defgroup CMSIS_glob_defs CMSIS Global Defines

 <strong>IO Type Qualifiers</strong> are used
 \li to specify the access to peripheral variables.
 \li for automatic generation of peripheral register debug information.
 */

#ifndef __cplusplus
#define __I volatile      /*!< Defines 'read only' permissions      */
#else
#define __I volatile const /*!< Defines 'read only' permissions      */

```

```
#endif

#define __O volatile /*!< Defines 'write only' permissions */
#define __IO volatile /*!< Defines 'read / write' permissions */

/*@} end of group Cortex_M7 */
```

```
*****
```

```
* Register Abstraction
```

```
Core Register contain:
```

- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core Debug Register
- Core MPU Register
- Core FPU Register

```
*****/
```

```
/** \defgroup CMSIS_core_register Defines and Type Definitions
 \brief Type definitions and defines for Cortex-M processor based devices.
 */


```

```
/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.

 @{
 */


```

```

/** \brief Union type to access the Application Program Status Register (APSR).
*/
typedef union
{
    struct
    {
        uint32_t _reserved0:16;           /*!< bit: 0..15 Reserved          */
        uint32_t GE:4;                  /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:7;           /*!< bit: 20..26 Reserved          */
        uint32_t Q:1;                  /*!< bit: 27 Saturation condition flag */
        uint32_t V:1;                  /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;                  /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;                  /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;                  /*!< bit: 31 Negative condition code flag */
    } b;                         /*!< Structure used for bit access */
    uint32_t w;                   /*!< Type used for word access   */
} APSR_Type;

```

```

/* APSR Register Definitions */

#define APSR_N_Pos      31           /*!< APSR: N Position */
#define APSR_N_Msk     (1UL << APSR_N_Pos)  /*!< APSR: N Mask */

#define APSR_Z_Pos      30           /*!< APSR: Z Position */
#define APSR_Z_Msk     (1UL << APSR_Z_Pos)  /*!< APSR: Z Mask */

#define APSR_C_Pos      29           /*!< APSR: C Position */
#define APSR_C_Msk     (1UL << APSR_C_Pos)  /*!< APSR: C Mask */

#define APSR_V_Pos      28           /*!< APSR: V Position */

```

```

#define APSR_V_Msk          (1UL << APSR_V_Pos)           /*!< APSR: V Mask */

#define APSR_Q_Pos           27                           /*!< APSR: Q Position */
#define APSR_Q_Msk          (1UL << APSR_Q_Pos)           /*!< APSR: Q Mask */

#define APSR_GE_Pos          16                           /*!< APSR: GE Position */
#define APSR_GE_Msk         (0xFUL << APSR_GE_Pos)        /*!< APSR: GE Mask */

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

```

*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} IPSR_Type;

```

/* IPSR Register Definitions */

```

#define IPSR_ISR_Pos          0           /*!< IPSR: ISR Position */
#define IPSR_ISR_Msk         (0x1FFUL /*<< IPSR_ISR_Pos*/)      /*!< IPSR: ISR Mask */

```

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).

```

*/
typedef union

```

```

{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:7;    /*!< bit: 9..15 Reserved */
        uint32_t GE:4;           /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:4;    /*!< bit: 20..23 Reserved */
        uint32_t T:1;             /*!< bit: 24 Thumb bit (read 0) */
        uint32_t IT:2;            /*!< bit: 25..26 saved IT state (read 0) */
        uint32_t Q:1;              /*!< bit: 27 Saturation condition flag */
        uint32_t V:1;              /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;              /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;              /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;              /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

```

/ xPSR Register Definitions */*

#define xPSR_N_Pos	31	/*!< xPSR: N Position */
#define xPSR_N_Msk	(1UL << xPSR_N_Pos)	/*!< xPSR: N Mask */
#define xPSR_Z_Pos	30	/*!< xPSR: Z Position */
#define xPSR_Z_Msk	(1UL << xPSR_Z_Pos)	/*!< xPSR: Z Mask */
#define xPSR_C_Pos	29	/*!< xPSR: C Position */
#define xPSR_C_Msk	(1UL << xPSR_C_Pos)	/*!< xPSR: C Mask */
#define xPSR_V_Pos	28	/*!< xPSR: V Position */

#define xPSR_V_Msk	(1UL << xPSR_V_Pos)	/*!< xPSR: V Mask */
#define xPSR_Q_Pos	27	/*!< xPSR: Q Position */
#define xPSR_Q_Msk	(1UL << xPSR_Q_Pos)	/*!< xPSR: Q Mask */
#define xPSR_IT_Pos	25	/*!< xPSR: IT Position */
#define xPSR_IT_Msk	(3UL << xPSR_IT_Pos)	/*!< xPSR: IT Mask */
#define xPSR_T_Pos	24	/*!< xPSR: T Position */
#define xPSR_T_Msk	(1UL << xPSR_T_Pos)	/*!< xPSR: T Mask */
#define xPSR_GE_Pos	16	/*!< xPSR: GE Position */
#define xPSR_GE_Msk	(0xFUL << xPSR_GE_Pos)	/*!< xPSR: GE Mask */
#define xPSR_ISR_Pos	0	/*!< xPSR: ISR Position */
#define xPSR_ISR_Msk	(0x1FFUL /*<< xPSR_ISR_Pos*/)	/*!< xPSR: ISR Mask */

```
/** \brief Union type to access the Control Registers (CONTROL).
 */
typedef union
{
    struct
    {
        uint32_t nPRIV:1;          /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t FPCA:1;           /*!< bit: 2 FP extension active flag */
        uint32_t _reserved0:29;     /*!< bit: 3..31 Reserved */
    } b;                         /*!< Structure used for bit access */
}
```

```

    uint32_t w;           /*!< Type used for word access */
} CONTROL_Type;

/* CONTROL Register Definitions */

#define CONTROL_FPCA_Pos      2           /*!< CONTROL: FPCA Position */
#define CONTROL_FPCA_Msk      (1UL << CONTROL_FPCA_Pos)      /*!< CONTROL: FPCA
Mask */

#define CONTROL_SPSEL_Pos     1           /*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk     (1UL << CONTROL_SPSEL_Pos)      /*!< CONTROL: SPSEL
Mask */

#define CONTROL_nPRIV_Pos     0           /*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk     (1UL /*<< CONTROL_nPRIV_Pos*/)      /*!< CONTROL:
nPRIV Mask */

/*@} end of group CMSIS_CORE */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
@{
*/

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[8];      /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */

```

```

    uint32_t RESERVED0[24];

    __IO uint32_t ICER[8];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
    uint32_t RSERVED1[24];

    __IO uint32_t ISPR[8];          /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */
    uint32_t RESERVED2[24];

    __IO uint32_t ICPR[8];          /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */
    uint32_t RESERVED3[24];

    __IO uint32_t IABR[8];          /*!< Offset: 0x200 (R/W) Interrupt Active bit Register */
    uint32_t RESERVED4[56];

    __IO uint8_t IP[240];           /*!< Offset: 0x300 (R/W) Interrupt Priority Register (8Bit wide) */

    uint32_t RESERVED5[644];

    __O uint32_t STIR;             /*!< Offset: 0xE00 ( /W) Software Trigger Interrupt Register */
} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */

#define NVIC_STIR_INTID_Pos        0                      /*!< STIR: INTLINESNUM Position */
#define NVIC_STIR_INTID_Msk        (0x1FFUL /*<< NVIC_STIR_INTID_Pos*/)      /*!< STIR:
INTLINESNUM Mask */

```

```
/*@} end of group CMSIS_NVIC */
```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers
@{
*/

```

```
/** \brief Structure type to access the System Control Block (SCB).
```

```

*/
typedef struct
{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ ) CPUID Base Register          */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
/*/
    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
    __IO uint32_t AIRCR;         /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control
Register   */
    __IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register          */
    __IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register    */
    __IO uint8_t SHPR[12];        /*!< Offset: 0x018 (R/W) System Handlers Priority Registers (4-7, 8-11,
12-15) */
    __IO uint32_t SHCSR;          /*!< Offset: 0x024 (R/W) System Handler Control and State Register
*/
    __IO uint32_t CFSR;           /*!< Offset: 0x028 (R/W) Configurable Fault Status Register */
/*/
    __IO uint32_t HFSR;           /*!< Offset: 0x02C (R/W) HardFault Status Register          */
    __IO uint32_t DFSR;           /*!< Offset: 0x030 (R/W) Debug Fault Status Register        */
    __IO uint32_t MMFAR;          /*!< Offset: 0x034 (R/W) MemManage Fault Address Register    */
/*/
    __IO uint32_t BFAR;           /*!< Offset: 0x038 (R/W) BusFault Address Register          */
    __IO uint32_t AFSR;           /*!< Offset: 0x03C (R/W) Auxiliary Fault Status Register    */
    __I uint32_t ID_PFR[2];       /*!< Offset: 0x040 (R/ ) Processor Feature Register          */
    __I uint32_t ID_DFR;          /*!< Offset: 0x048 (R/ ) Debug Feature Register            */
    __I uint32_t ID_AFR;          /*!< Offset: 0x04C (R/ ) Auxiliary Feature Register         */
    __I uint32_t ID_MFR[4];       /*!< Offset: 0x050 (R/ ) Memory Model Feature Register     */
/*/
    __I uint32_t ID_ISAR[5];      /*!< Offset: 0x060 (R/ ) Instruction Set Attributes Register
    uint32_t RESERVED0[1];
    __I uint32_t CLIDR;           /*!< Offset: 0x078 (R/ ) Cache Level ID register          */
*/
}

```

```

__I uint32_t CTR;           /*!< Offset: 0x07C (R/ ) Cache Type register          */
__I uint32_t CCSIDR;        /*!< Offset: 0x080 (R/ ) Cache Size ID Register      */
__IO uint32_t CSSELR;       /*!< Offset: 0x084 (R/W) Cache Size Selection Register */
__IO uint32_t CPACR;        /*!< Offset: 0x088 (R/W) Coprocessor Access Control Register */
*/
uint32_t RESERVED3[93];

__O uint32_t STIR;         /*!< Offset: 0x200 ( /W) Software Triggered Interrupt Register */
*/
uint32_t RESERVED4[15];

__I uint32_t MVFR0;        /*!< Offset: 0x240 (R/ ) Media and VFP Feature Register 0 */
*/
__I uint32_t MVFR1;        /*!< Offset: 0x244 (R/ ) Media and VFP Feature Register 1 */
*/
__I uint32_t MVFR2;        /*!< Offset: 0x248 (R/ ) Media and VFP Feature Register 1 */
*/
uint32_t RESERVED5[1];

__O uint32_t ICIALLU;      /*!< Offset: 0x250 ( /W) I-Cache Invalidate All to PoU      */
uint32_t RESERVED6[1];

__O uint32_t ICIMVAU;      /*!< Offset: 0x258 ( /W) I-Cache Invalidate by MVA to PoU */
*/
__O uint32_t DCIMVAC;      /*!< Offset: 0x25C ( /W) D-Cache Invalidate by MVA to PoC */
*/
__O uint32_t DCISW;         /*!< Offset: 0x260 ( /W) D-Cache Invalidate by Set-way */
*/
__O uint32_t DCCMVAU;      /*!< Offset: 0x264 ( /W) D-Cache Clean by MVA to PoU */
*/
__O uint32_t DCCMVAC;      /*!< Offset: 0x268 ( /W) D-Cache Clean by MVA to PoC */
*/
__O uint32_t DCCSW;         /*!< Offset: 0x26C ( /W) D-Cache Clean by Set-way          */
__O uint32_t DCCIMVAC;     /*!< Offset: 0x270 ( /W) D-Cache Clean and Invalidate by MVA to
PoC      */
*/
__O uint32_t DCCISW;        /*!< Offset: 0x274 ( /W) D-Cache Clean and Invalidate by Set-way */
*/

```

```

    uint32_t RESERVED7[6];

    __IO uint32_t ITCMCR;           /*!< Offset: 0x290 (R/W) Instruction Tightly-Coupled Memory Control
Register */

    __IO uint32_t DTCMCR;           /*!< Offset: 0x294 (R/W) Data Tightly-Coupled Memory Control
Registers */

    __IO uint32_t AHBPCR;           /*!< Offset: 0x298 (R/W) AHB Control Register */

    __IO uint32_t CACR;             /*!< Offset: 0x29C (R/W) L1 Cache Control Register */

    __IO uint32_t AHBSCR;           /*!< Offset: 0x2A0 (R/W) AHB Slave Control Register
*/
}

uint32_t RESERVED8[1];

__IO uint32_t ABFSR;             /*!< Offset: 0x2A8 (R/W) Auxiliary Bus Fault Status Register
*/
}

} SCB_Type;

```

```

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24          /*!< SCB CPUID:
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk     (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos) /*!<
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos        20          /*!< SCB CPUID: VARIANT Position
*/

#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos) /*!< SCB CPUID:
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos   16          /*!< SCB CPUID:
ARCHITECTURE Position */

#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos) /*!<
SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos         4           /*!< SCB CPUID: PARTNO Position
*/

```

```

#define SCB_CPUID_PARTNO_Msk      (0xFFFFUL << SCB_CPUID_PARTNO_Pos)      /*!< SCB
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos    0                                     /*!< SCB CPUID: REVISION Position */
/*/

#define SCB_CPUID_REVISION_Msk    (0xFUL /*<< SCB_CPUID_REVISION_Pos*/)      /*!< SCB
CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos   31                                    /*!< SCB ICSR: NMIPENDSET
Position */

#define SCB_ICSR_NMIPENDSET_Msk   (1UL << SCB_ICSR_NMIPENDSET_Pos)      /*!< SCB ICSR:
NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos    28                                    /*!< SCB ICSR: PENDSVSET
Position */

#define SCB_ICSR_PENDSVSET_Msk   (1UL << SCB_ICSR_PENDSVSET_Pos)      /*!< SCB ICSR:
PENDSVSET Mask */

#define SCB_ICSR_PENDSVCLR_Pos   27                                    /*!< SCB ICSR: PENDSVCLR
Position */

#define SCB_ICSR_PENDSVCLR_Msk   (1UL << SCB_ICSR_PENDSVCLR_Pos)      /*!< SCB ICSR:
PENDSVCLR Mask */

#define SCB_ICSR_PENDSTSET_Pos   26                                    /*!< SCB ICSR: PENDSTSET Position
*/
#define SCB_ICSR_PENDSTSET_Msk   (1UL << SCB_ICSR_PENDSTSET_Pos)      /*!< SCB ICSR:
PENDSTSET Mask */

#define SCB_ICSR_PENDSTCLR_Pos   25                                    /*!< SCB ICSR: PENDSTCLR
Position */

#define SCB_ICSR_PENDSTCLR_Msk   (1UL << SCB_ICSR_PENDSTCLR_Pos)      /*!< SCB ICSR:
PENDSTCLR Mask */

```

```

#define SCB_ICSR_ISRPREEMPT_Pos      23          /*!< SCB ICSR: ISRPREEMPT
Position */

#define SCB_ICSR_ISRPREEMPT_Msk      (1UL << SCB_ICSR_ISRPREEMPT_Pos)    /*!< SCB ICSR:
ISRPREEMPT Mask */

#define SCB_ICSR_ISRPENDING_Pos     22          /*!< SCB ICSR: ISR PENDING
Position */

#define SCB_ICSR_ISRPENDING_Msk      (1UL << SCB_ICSR_ISRPENDING_Pos)    /*!< SCB ICSR:
ISR PENDING Mask */

#define SCB_ICSR_VECTPENDING_Pos    12          /*!< SCB ICSR: VECTPENDING
Position */

#define SCB_ICSR_VECTPENDING_Msk      (0x1FFUL << SCB_ICSR_VECTPENDING_Pos)  /*!< SCB
ICSR: VECTPENDING Mask */

#define SCB_ICSR_RETTOBASE_Pos       11          /*!< SCB ICSR: RETTOBASE
Position */

#define SCB_ICSR_RETTOBASE_Msk      (1UL << SCB_ICSR_RETTOBASE_Pos)    /*!< SCB ICSR:
RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos      0           /*!< SCB ICSR: VECTACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk      (0x1FFFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/) /*!< SCB
ICSR: VECTACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */

#define SCB_VTOR_TBLOFF_Pos        7           /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk      (0xFFFFFFFFUL << SCB_VTOR_TBLOFF_Pos)  /*!< SCB
VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */

```

#define SCB_AIRCR_VECTKEY_Pos	16	/*!< SCB AIRCR: VECTKEY Position */
#define SCB_AIRCR_VECTKEY_Msk	(0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)	/*!< SCB AIRCR: VECTKEY Mask */
#define SCB_AIRCR_VECTKEYSTAT_Pos	16	/*!< SCB AIRCR: VECTKEYSTAT Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk	(0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)	/*!< SCB AIRCR: VECTKEYSTAT Mask */
#define SCB_AIRCR_ENDIANESS_Pos	15	/*!< SCB AIRCR: ENDIANESS Position */
#define SCB_AIRCR_ENDIANESS_Msk	(1UL << SCB_AIRCR_ENDIANESS_Pos)	/*!< SCB AIRCR: ENDIANESS Mask */
#define SCB_AIRCR_PRIGROUP_Pos	8	/*!< SCB AIRCR: PRIGROUP Position */
#define SCB_AIRCR_PRIGROUP_Msk	(7UL << SCB_AIRCR_PRIGROUP_Pos)	/*!< SCB AIRCR: PRIGROUP Mask */
#define SCB_AIRCR_SYSRESETREQ_Pos	2	/*!< SCB AIRCR: SYSRESETREQ Position */
#define SCB_AIRCR_SYSRESETREQ_Msk	(1UL << SCB_AIRCR_SYSRESETREQ_Pos)	/*!< SCB AIRCR: SYSRESETREQ Mask */
#define SCB_AIRCR_VECTCLRACTIVE_Pos	1	/*!< SCB AIRCR: VECTCLRACTIVE Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk	(1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)	/*!< SCB AIRCR: VECTCLRACTIVE Mask */
#define SCB_AIRCR_VECTRESET_Pos	0	/*!< SCB AIRCR: VECTRESET Position */
#define SCB_AIRCR_VECTRESET_Msk	(1UL /*<< SCB_AIRCR_VECTRESET_Pos*/)	/*!< SCB AIRCR: VECTRESET Mask */

```

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos      4          /*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk      (1UL << SCB_SCR_SEVONPEND_Pos)      /*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos     2          /*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk      (1UL << SCB_SCR_SLEEPDEEP_Pos)      /*!< SCB SCR: SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos    1          /*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk    (1UL << SCB_SCR_SLEEPONEXIT_Pos)      /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_BP_Pos           18         /*!< SCB CCR: Branch prediction enable bit Position */
#define SCB_CCR_BP_Msk            (1UL << SCB_CCR_BP_Pos)      /*!< SCB CCR: Branch prediction enable bit Mask */

#define SCB_CCR_IC_Pos           17         /*!< SCB CCR: Instruction cache enable bit Position */
#define SCB_CCR_IC_Msk            (1UL << SCB_CCR_IC_Pos)      /*!< SCB CCR: Instruction cache enable bit Mask */

#define SCB_CCR_DC_Pos           16         /*!< SCB CCR: Cache enable bit Position */
#define SCB_CCR_DC_Msk            (1UL << SCB_CCR_DC_Pos)      /*!< SCB CCR: Cache enable bit Mask */

```

```

#define SCB_CCR_STKALIGN_Pos      9          /*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk      (1UL << SCB_CCR_STKALIGN_Pos)      /*!< SCB CCR:
STKALIGN Mask */

#define SCB_CCR_BFHFMIGN_Pos      8          /*!< SCB CCR: BFHFMIGN
Position */
#define SCB_CCR_BFHFMIGN_Msk      (1UL << SCB_CCR_BFHFMIGN_Pos)      /*!< SCB CCR:
BFHFMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos      4          /*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk      (1UL << SCB_CCR_DIV_0_TRP_Pos)      /*!< SCB CCR:
DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos      3          /*!< SCB CCR: UNALIGN_TRP
Position */
#define SCB_CCR_UNALIGN_TRP_Msk      (1UL << SCB_CCR_UNALIGN_TRP_Pos)      /*!< SCB
CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos      1          /*!< SCB CCR: USERSETMPEND
Position */
#define SCB_CCR_USERSETMPEND_Msk      (1UL << SCB_CCR_USERSETMPEND_Pos)      /*!< SCB
CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos      0          /*!< SCB CCR:
NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDENA_Msk      (1UL /*<< SCB_CCR_NONBASETHRDENA_Pos*/)      /*!< SCB CCR:
NONBASETHRDENA Mask */

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_USGFAULTENA_Pos      18         /*!< SCB SHCSR:
USGFAULTENA Position */

```

```
#define SCB_SHCSR_USGFAULTENA_Msk      (1UL << SCB_SHCSR_USGFAULTENA_Pos)      /*!< SCB
SHCSR: USGFAULTENA Mask */

#define SCB_SHCSR_BUSFAULTENA_Pos       17                                /*!< SCB SHCSR: BUSFAULTENA
Position */

#define SCB_SHCSR_BUSFAULTENA_Msk      (1UL << SCB_SHCSR_BUSFAULTENA_Pos)      /*!< SCB
SHCSR: BUSFAULTENA Mask */

#define SCB_SHCSR_MEMFAULTENA_Pos      16                                /*!< SCB SHCSR:
MEMFAULTENA Position */

#define SCB_SHCSR_MEMFAULTENA_Msk      (1UL << SCB_SHCSR_MEMFAULTENA_Pos)      /*!<
SCB SHCSR: MEMFAULTENA Mask */

#define SCB_SHCSR_SVCALLPENDED_Pos     15                                /*!< SCB SHCSR:
SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk     (1UL << SCB_SHCSR_SVCALLPENDED_Pos)      /*!<
SCB SHCSR: SVCALLPENDED Mask */

#define SCB_SHCSR_BUSFAULTPENDED_Pos   14                                /*!< SCB SHCSR:
BUSFAULTPENDED Position */

#define SCB_SHCSR_BUSFAULTPENDED_Msk   (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)      /*!<
SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos   13                                /*!< SCB SHCSR:
MEMFAULTPENDED Position */

#define SCB_SHCSR_MEMFAULTPENDED_Msk   (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)      /*!<
SCB SHCSR: MEMFAULTPENDED Mask */

#define SCB_SHCSR_USGFAULTPENDED_Pos   12                                /*!< SCB SHCSR:
USGFAULTPENDED Position */

#define SCB_SHCSR_USGFAULTPENDED_Msk   (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)      /*!<
SCB SHCSR: USGFAULTPENDED Mask */
```

#define SCB_SHCSR_SYSTICKACT_Pos	11	/*!< SCB SHCSR: SYSTICKACT Position */
#define SCB_SHCSR_SYSTICKACT_Msk	(1UL << SCB_SHCSR_SYSTICKACT_Pos)	/*!< SCB SHCSR: SYSTICKACT Mask */
#define SCB_SHCSR_PENDSVACT_Pos	10	/*!< SCB SHCSR: PENDSVACT Position */
#define SCB_SHCSR_PENDSVACT_Msk	(1UL << SCB_SHCSR_PENDSVACT_Pos)	/*!< SCB SHCSR: PENDSVACT Mask */
#define SCB_SHCSR_MONITORACT_Pos	8	/*!< SCB SHCSR: MONITORACT Position */
#define SCB_SHCSR_MONITORACT_Msk	(1UL << SCB_SHCSR_MONITORACT_Pos)	/*!< SCB SHCSR: MONITORACT Mask */
#define SCB_SHCSR_SVCALLACT_Pos	7	/*!< SCB SHCSR: SVCALLACT Position */
#define SCB_SHCSR_SVCALLACT_Msk	(1UL << SCB_SHCSR_SVCALLACT_Pos)	/*!< SCB SHCSR: SVCALLACT Mask */
#define SCB_SHCSR_USGFAULTACT_Pos	3	/*!< SCB SHCSR: USGFAULTACT Position */
#define SCB_SHCSR_USGFAULTACT_Msk	(1UL << SCB_SHCSR_USGFAULTACT_Pos)	/*!< SCB SHCSR: USGFAULTACT Mask */
#define SCB_SHCSR_BUSFAULTACT_Pos	1	/*!< SCB SHCSR: BUSFAULTACT Position */
#define SCB_SHCSR_BUSFAULTACT_Msk	(1UL << SCB_SHCSR_BUSFAULTACT_Pos)	/*!< SCB SHCSR: BUSFAULTACT Mask */
#define SCB_SHCSR_MEMFAULTACT_Pos	0	/*!< SCB SHCSR: MEMFAULTACT Position */
#define SCB_SHCSR_MEMFAULTACT_Msk	(1UL /*<< SCB_SHCSR_MEMFAULTACT_Pos*/)	/*!< SCB SHCSR: MEMFAULTACT Mask */

```

/* SCB Configurable Fault Status Registers Definitions */

#define SCB_CFSR_USGFAULTSR_Pos      16          /*!< SCB CFSR: Usage Fault
Status Register Position */

#define SCB_CFSR_USGFAULTSR_Msk      (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos)  /*!< SCB
CFSR: Usage Fault Status Register Mask */

#define SCB_CFSR_BUSFAULTSR_Pos      8           /*!< SCB CFSR: Bus Fault Status
Register Position */

#define SCB_CFSR_BUSFAULTSR_Msk      (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos)  /*!< SCB
CFSR: Bus Fault Status Register Mask */

#define SCB_CFSR_MEMFAULTSR_Pos      0           /*!< SCB CFSR: Memory Manage
Fault Status Register Position */

#define SCB_CFSR_MEMFAULTSR_Msk      (0xFFUL /*<< SCB_CFSR_MEMFAULTSR_Pos*/)  /*!<
SCB CFSR: Memory Manage Fault Status Register Mask */

/* SCB Hard Fault Status Registers Definitions */

#define SCB_HFSR_DEBUGEVT_Pos        31          /*!< SCB HFSR: DEBUGEVT
Position */

#define SCB_HFSR_DEBUGEVT_Msk        (1UL << SCB_HFSR_DEBUGEVT_Pos)  /*!< SCB HFSR:
DEBUGEVT Mask */

#define SCB_HFSR_FORCED_Pos         30          /*!< SCB HFSR: FORCED Position */

#define SCB_HFSR_FORCED_Msk         (1UL << SCB_HFSR_FORCED_Pos)  /*!< SCB HFSR:
FORCED Mask */

#define SCB_HFSR_VECTTBL_Pos         1           /*!< SCB HFSR: VECTTBL Position */

#define SCB_HFSR_VECTTBL_Msk         (1UL << SCB_HFSR_VECTTBL_Pos)  /*!< SCB HFSR:
VECTTBL Mask */

/* SCB Debug Fault Status Register Definitions */

```

```

#define SCB_DFSR_EXTERNAL_Pos      4          /*!< SCB DFSR: EXTERNAL Position */
*/
#define SCB_DFSR_EXTERNAL_Msk     (1UL << SCB_DFSR_EXTERNAL_Pos)      /*!< SCB DFSR:
EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos       3          /*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk      (1UL << SCB_DFSR_VCATCH_Pos)      /*!< SCB DFSR:
VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos     2          /*!< SCB DFSR: DWTTRAP Position */
*/
#define SCB_DFSR_DWTTRAP_Msk    (1UL << SCB_DFSR_DWTTRAP_Pos)      /*!< SCB DFSR:
DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos        1          /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk       (1UL << SCB_DFSR_BKPT_Pos)      /*!< SCB DFSR: BKPT
Mask */

#define SCB_DFSR_HALTED_Pos      0          /*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk     (1UL /*<< SCB_DFSR_HALTED_Pos*/)      /*!< SCB DFSR:
HALTED Mask */

/* Cache Level ID register */
#define SCB_CLIDR_LOUU_Pos       27         /*!< SCB CLIDR: LoUU Position */
#define SCB_CLIDR_LOUU_Msk      (7UL << SCB_CLIDR_LOUU_Pos)      /*!< SCB CLIDR:
LoUU Mask */

#define SCB_CLIDR_LOC_Pos        24         /*!< SCB CLIDR: LoC Position */
#define SCB_CLIDR_LOC_Msk       (7UL << SCB_CLIDR_FORMAT_Pos)      /*!< SCB CLIDR: LoC
Mask */

/* Cache Type register */

```

```

#define SCB_CTR_FORMAT_Pos      29          /*!< SCB CTR: Format Position */
#define SCB_CTR_FORMAT_Msk      (7UL << SCB_CTR_FORMAT_Pos)      /*!< SCB CTR:
Format Mask */

#define SCB_CTR_CWG_Pos        24          /*!< SCB CTR: CWG Position */
#define SCB_CTR_CWG_Msk        (0xFUL << SCB_CTR_CWG_Pos)      /*!< SCB CTR: CWG
Mask */

#define SCB_CTR_ERG_Pos        20          /*!< SCB CTR: ERG Position */
#define SCB_CTR_ERG_Msk        (0xFUL << SCB_CTR_ERG_Pos)      /*!< SCB CTR: ERG
Mask */

#define SCB_CTR_DMINLINE_Pos    16          /*!< SCB CTR: DminLine Position */
#define SCB_CTR_DMINLINE_Msk    (0xFUL << SCB_CTR_DMINLINE_Pos)      /*!< SCB CTR:
DminLine Mask */

#define SCB_CTR_IMINLINE_Pos    0           /*!< SCB CTR: ImInLine Position */
#define SCB_CTR_IMINLINE_Msk    (0xFUL /*<< SCB_CTR_IMINLINE_Pos*/)      /*!< SCB CTR:
ImInLine Mask */

/* Cache Size ID Register */

#define SCB_CCSIDR_WT_Pos       31          /*!< SCB CCSIDR: WT Position */
#define SCB_CCSIDR_WT_Msk       (7UL << SCB_CCSIDR_WT_Pos)      /*!< SCB CCSIDR: WT
Mask */

#define SCB_CCSIDR_WB_Pos       30          /*!< SCB CCSIDR: WB Position */
#define SCB_CCSIDR_WB_Msk       (7UL << SCB_CCSIDR_WB_Pos)      /*!< SCB CCSIDR: WB
Mask */

#define SCB_CCSIDR_RA_Pos       29          /*!< SCB CCSIDR: RA Position */
#define SCB_CCSIDR_RA_Msk       (7UL << SCB_CCSIDR_RA_Pos)      /*!< SCB CCSIDR: RA
Mask */

```

```

#define SCB_CCSIDR_WA_Pos           28          /*!< SCB CCSIDR: WA Position */
#define SCB_CCSIDR_WA_Msk          (7UL << SCB_CCSIDR_WA_Pos)      /*!< SCB CCSIDR: WA
Mask */

#define SCB_CCSIDR_NUMSETS_Pos     13          /*!< SCB CCSIDR: NumSets
Position */
#define SCB_CCSIDR_NUMSETS_Msk    (0x7FFFUL << SCB_CCSIDR_NUMSETS_Pos)  /*!< SCB
CCSIDR: NumSets Mask */

#define SCB_CCSIDR_ASSOCIATIVITY_Pos 3          /*!< SCB CCSIDR: Associativity
Position */
#define SCB_CCSIDR_ASSOCIATIVITY_Msk (0x3FFUL << SCB_CCSIDR_ASSOCIATIVITY_Pos) /*!< SCB
CCSIDR: Associativity Mask */

#define SCB_CCSIDR_LINESIZE_Pos    0          /*!< SCB CCSIDR: LineSize Position
*/
#define SCB_CCSIDR_LINESIZE_Msk   (7UL /*<< SCB_CCSIDR_LINESIZE_Pos*/)  /*!< SCB
CCSIDR: LineSize Mask */

/* Cache Size Selection Register */
#define SCB_CSSELR_LEVEL_Pos       1          /*!< SCB CSSELR: Level Position */
#define SCB_CSSELR_LEVEL_Msk      (7UL << SCB_CSSELR_LEVEL_Pos)      /*!< SCB CSSELR:
Level Mask */

#define SCB_CSSELR_IND_Pos        0          /*!< SCB CSSELR: InD Position */
#define SCB_CSSELR_IND_Msk       (1UL /*<< SCB_CSSELR_IND_Pos*/)  /*!< SCB CSSELR:
InD Mask */

/* SCB Software Triggered Interrupt Register */
#define SCB_STIR_INTID_Pos        0          /*!< SCB STIR: INTID Position */

```

```

#define SCB_STIR_INTID_Msk      (0x1FFUL /*<< SCB_STIR_INTID_Pos*/)    /*!< SCB STIR: INTID
Mask */

/* Instruction Tightly-Coupled Memory Control Register*/

#define SCB_ITCMCR_SZ_Pos       3                                /*!< SCB ITCMCR: SZ Position */
#define SCB_ITCMCR_SZ_Msk       (0xFUL << SCB_ITCMCR_SZ_Pos)        /*!< SCB ITCMCR: SZ
Mask */

#define SCB_ITCMCR_RETEN_Pos    2                                /*!< SCB ITCMCR: RETEN Position
*/
#define SCB_ITCMCR_RETEN_Msk   (1UL << SCB_ITCMCR_RETEN_Pos)        /*!< SCB ITCMCR:
RETEN Mask */

#define SCB_ITCMCR_RMW_Pos      1                                /*!< SCB ITCMCR: RMW Position */
#define SCB_ITCMCR_RMW_Msk     (1UL << SCB_ITCMCR_RMW_Pos)        /*!< SCB
ITCMCR: RMW Mask */

#define SCB_ITCMCR_EN_Pos       0                                /*!< SCB ITCMCR: EN Position */
#define SCB_ITCMCR_EN_Msk      (1UL /*<< SCB_ITCMCR_EN_Pos*/)        /*!< SCB ITCMCR:
EN Mask */

/* Data Tightly-Coupled Memory Control Registers*/

#define SCB_DTCMCR_SZ_Pos       3                                /*!< SCB DTCMCR: SZ Position */
#define SCB_DTCMCR_SZ_Msk       (0xFUL << SCB_DTCMCR_SZ_Pos)        /*!< SCB DTCMCR:
SZ Mask */

#define SCB_DTCMCR_RETEN_Pos    2                                /*!< SCB DTCMCR: RETEN Position
*/
#define SCB_DTCMCR_RETEN_Msk   (1UL << SCB_DTCMCR_RETEN_Pos)        /*!< SCB
DTCMCR: RETEN Mask */

```

```

#define SCB_DTCMCR_RMW_Pos      1          /*!< SCB DTCMCR: RMW Position */
*/
#define SCB_DTCMCR_RMW_Msk      (1UL << SCB_DTCMCR_RMW_Pos)      /*!< SCB
DTCMCR: RMW Mask */

#define SCB_DTCMCR_EN_Pos       0          /*!< SCB DTCMCR: EN Position */
#define SCB_DTCMCR_EN_Msk       (1UL /*<< SCB_DTCMCR_EN_Pos*/)      /*!< SCB
DTCMCR: EN Mask */

/* AHB Control Register */
#define SCB_AHBPCR_SZ_Pos       1          /*!< SCB AHBPCR: SZ Position */
#define SCB_AHBPCR_SZ_Msk       (7UL << SCB_AHBPCR_SZ_Pos)      /*!< SCB AHBPCR: SZ
Mask */

#define SCB_AHBPCR_EN_Pos       0          /*!< SCB AHBPCR: EN Position */
#define SCB_AHBPCR_EN_Msk       (1UL /*<< SCB_AHBPCR_EN_Pos*/)      /*!< SCB AHBPCR:
EN Mask */

/* L1 Cache Control Register */
#define SCB_CACR_FORCEWT_Pos     2          /*!< SCB CACR: FORCEWT Position */
*/
#define SCB_CACR_FORCEWT_Msk     (1UL << SCB_CACR_FORCEWT_Pos)      /*!< SCB CACR:
FORCEWT Mask */

#define SCB_CACR_ECCEN_Pos       1          /*!< SCB CACR: ECCEN Position */
#define SCB_CACR_ECCEN_Msk       (1UL << SCB_CACR_ECCEN_Pos)      /*!< SCB CACR:
ECCEN Mask */

#define SCB_CACR_SIWT_Pos       0          /*!< SCB CACR: SIWT Position */
#define SCB_CACR_SIWT_Msk       (1UL /*<< SCB_CACR_SIWT_Pos*/)      /*!< SCB CACR:
SIWT Mask */

```

```

/* AHBS control register */

#define SCB_AHBSCR_INITCOUNT_Pos      11          /*!< SCB AHBSCR: INITCOUNT Position */
#define SCB_AHBSCR_INITCOUNT_Msk      (0x1FUL << SCB_AHBPCR_INITCOUNT_Pos)    /*!< SCB AHBSCR: INITCOUNT Mask */

#define SCB_AHBSCR_TPRI_Pos          2           /*!< SCB AHBSCR: TPRI Position */
#define SCB_AHBSCR_TPRI_Msk          (0x1FFUL << SCB_AHBPCR_TPRI_Pos)      /*!< SCB AHBSCR: TPRI Mask */

#define SCB_AHBSCR_CTL_Pos           0           /*!< SCB AHBSCR: CTL Position*/
#define SCB_AHBSCR_CTL_Msk           (3UL /*<< SCB_AHBPCR_CTL_Pos*/)      /*!< SCB AHBSCR: CTL Mask */

/* Auxiliary Bus Fault Status Register */

#define SCB_ABFSR_AXIMTYPE_Pos       8           /*!< SCB ABFSR: AXIMTYPE Position*/
#define SCB_ABFSR_AXIMTYPE_Msk       (3UL << SCB_ABFSR_AXIMTYPE_Pos)      /*!< SCB ABFSR: AXIMTYPE Mask */

#define SCB_ABFSR_EPPB_Pos           4           /*!< SCB ABFSR: EPPB Position*/
#define SCB_ABFSR_EPPB_Msk           (1UL << SCB_ABFSR_EPPB_Pos)        /*!< SCB ABFSR: EPPB Mask */

#define SCB_ABFSR_AXIM_Pos           3           /*!< SCB ABFSR: AXIM Position*/
#define SCB_ABFSR_AXIM_Msk           (1UL << SCB_ABFSR_AXIM_Pos)        /*!< SCB ABFSR: AXIM Mask */

#define SCB_ABFSR_AHBP_Pos           2           /*!< SCB ABFSR: AHBP Position*/
#define SCB_ABFSR_AHBP_Msk           (1UL << SCB_ABFSR_AHBP_Pos)        /*!< SCB ABFSR: AHBP Mask */

```

```

#define SCB_ABFSR_DTCM_Pos      1                  /*!< SCB ABFSR: DTCM Position*/
#define SCB_ABFSR_DTCM_Msk      (1UL << SCB_ABFSR_DTCM_Pos)    /*!< SCB ABFSR:
DTCM Mask */

#define SCB_ABFSR_ITCM_Pos      0                  /*!< SCB ABFSR: ITCM Position*/
#define SCB_ABFSR_ITCM_Msk      (1UL /*<< SCB_ABFSR_ITCM_Pos*/) /*!< SCB ABFSR:
ITCM Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
\brief Type definitions for the System Control and ID Register not in the SCB
@{
*/

/** \brief Structure type to access the System Control and ID Register not in the SCB.
*/
typedef struct
{
    uint32_t RESERVED0[1];
    __I uint32_t ICTR;           /*!< Offset: 0x004 (R/ ) Interrupt Controller Type Register */
    __IO uint32_t ACTLR;        /*!< Offset: 0x008 (R/W) Auxiliary Control Register */
} SCnSCB_Type;

/* Interrupt Controller Type Register Definitions */

#define SCnSCB_ICTR_INTLINESNUM_Pos 0          /*!< ICTR: INTLINESNUM Position */

```

```

#define SCnSCB_ICTR_INTLINESNUM_Msk      (0xFUL /*<< SCnSCB_ICTR_INTLINESNUM_Pos*/) /*!<
ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define SCnSCB_ACTLR_DISITMATBFLUSH_Pos  12          /*!< ACTLR: DISITMATBFLUSH
Position */

#define SCnSCB_ACTLR_DISITMATBFLUSH_Msk  (1UL << SCnSCB_ACTLR_DISITMATBFLUSH_Pos) /*!<
ACTLR: DISITMATBFLUSH Mask */

#define SCnSCB_ACTLR_DISRAMODE_Pos     11          /*!< ACTLR: DISRAMODE
Position */

#define SCnSCB_ACTLR_DISRAMODE_Msk    (1UL << SCnSCB_ACTLR_DISRAMODE_Pos) /*!<
ACTLR: DISRAMODE Mask */

#define SCnSCB_ACTLR_FPEXCODIS_Pos   10          /*!< ACTLR: FPEXCODIS Position */
*/

#define SCnSCB_ACTLR_FPEXCODIS_Msk  (1UL << SCnSCB_ACTLR_FPEXCODIS_Pos) /*!< ACTLR:
FPEXCODIS Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos     2           /*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk   (1UL << SCnSCB_ACTLR_DISFOLD_Pos) /*!< ACTLR:
DISFOLD Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos  0           /*!< ACTLR: DISMCYCINT
Position */

#define SCnSCB_ACTLR_DISMCYCINT_Msk (1UL /*<< SCnSCB_ACTLR_DISMCYCINT_Pos*/) /*!<
ACTLR: DISMCYCINT Mask */

/*@} end of group CMSIS_SCnotSCB */

```

```
/** \ingroup CMSIS_core_register
```

```

\defgroup CMSIS_SysTick System Tick Timer (SysTick)

\brief Type definitions for the System Timer Registers.

@{

/*
** \brief Structure type to access the System Timer (SysTick).
*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I  uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
} SysTick_Type;

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos     16          /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk     (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2           /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk     (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos      1           /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk      (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

```

```

#define SysTick_CTRL_ENABLE_Pos      0           /*!< SysTick CTRL: ENABLE Position */
/*/

#define SysTick_CTRL_ENABLE_Msk     (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)    /*!< SysTick
CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos     0           /*!< SysTick LOAD: RELOAD
Position */

#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/) /*!<
SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos     0           /*!< SysTick VAL: CURRENT
Position */

#define SysTick_VAL_CURRENT_Msk     (0xFFFFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/) /*!<
SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos     31          /*!< SysTick CALIB: NOREF Position */
/*/

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)        /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos       30          /*!< SysTick CALIB: SKEW Position */
/*/

#define SysTick_CALIB_SKEW_Msk       (1UL << SysTick_CALIB_SKEW_Pos)        /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos      0           /*!< SysTick CALIB: TENMS Position */
/*/

#define SysTick_CALIB_TENMS_Msk     (0xFFFFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

```

```

/*@} end of group CMSIS_SysTick */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_ITM Instrumentation Trace Macrocell (ITM)
 \brief Type definitions for the Instrumentation Trace Macrocell (ITM)

 @{
 */

/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
 */
typedef struct
{
    __O union
    {
        __O uint8_t u8;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit      */
        __O uint16_t u16;        /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit     */
        __O uint32_t u32;        /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit     */
    } PORT [32];             /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers   */

    uint32_t RESERVED0[864];

    __IO uint32_t TER;        /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register    */
    uint32_t RESERVED1[15];

    __IO uint32_t TPR;        /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register */
    uint32_t RESERVED2[15];

    __IO uint32_t TCR;        /*!< Offset: 0xE80 (R/W) ITM Trace Control Register   */
    uint32_t RESERVED3[29];

    __O uint32_t IWR;         /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register */
    __I uint32_t IRR;         /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register */
    __IO uint32_t IMCR;       /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
}

```

```

    uint32_t RESERVED4[43];

    __O uint32_t LAR;           /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register      */
    __I uint32_t LSR;           /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register       */
    uint32_t RESERVED5[6];

    __I uint32_t PID4;          /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
    __I uint32_t PID5;          /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
    __I uint32_t PID6;          /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
    __I uint32_t PID7;          /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
    __I uint32_t PID0;          /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
    __I uint32_t PID1;          /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
    __I uint32_t PID2;          /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
    __I uint32_t PID3;          /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
    __I uint32_t CID0;          /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
    __I uint32_t CID1;          /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
    __I uint32_t CID2;          /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */
    __I uint32_t CID3;          /*!< Offset: 0xFFC (R/ ) ITM Component Identification Register #3 */

} ITM_Type;

```

```

/* ITM Trace Privilege Register Definitions */

#define ITM_TPR_PRIVMASK_Pos      0                  /*!< ITM TPR: PRIVMASK Position */
#define ITM_TPR_PRIVMASK_Msk      (0xFUL /*<< ITM_TPR_PRIVMASK_Pos*/)      /*!< ITM TPR: PRIVMASK Mask */

```

```

/* ITM Trace Control Register Definitions */

#define ITM_TCR_BUSY_Pos         23                /*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk         (1UL << ITM_TCR_BUSY_Pos)        /*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos   16                /*!< ITM TCR: ATBID Position */

```

#define ITM_TCR_TraceBusID_Msk	(0x7FUL << ITM_TCR_TraceBusID_Pos)	/*!< ITM TCR: ATBID Mask */
#define ITM_TCR_GTSFREQ_Pos	10	/*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk	(3UL << ITM_TCR_GTSFREQ_Pos)	/*!< ITM TCR: Global timestamp frequency Mask */
#define ITM_TCR_TSPrescale_Pos	8	/*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk	(3UL << ITM_TCR_TSPrescale_Pos)	/*!< ITM TCR: TSPrescale Mask */
#define ITM_TCR_SWOENA_Pos	4	/*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk	(1UL << ITM_TCR_SWOENA_Pos)	/*!< ITM TCR: SWOENA Mask */
#define ITM_TCR_DWTENA_Pos	3	/*!< ITM TCR: DWTENA Position */
#define ITM_TCR_DWTENA_Msk	(1UL << ITM_TCR_DWTENA_Pos)	/*!< ITM TCR: DWTENA Mask */
#define ITM_TCR_SYNCENA_Pos	2	/*!< ITM TCR: SYNCENA Position */
#define ITM_TCR_SYNCENA_Msk	(1UL << ITM_TCR_SYNCENA_Pos)	/*!< ITM TCR: SYNCENA Mask */
#define ITM_TCR_TSENA_Pos	1	/*!< ITM TCR: TSENA Position */
#define ITM_TCR_TSENA_Msk	(1UL << ITM_TCR_TSENA_Pos)	/*!< ITM TCR: TSENA Mask */
#define ITM_TCR_ITMENA_Pos	0	/*!< ITM TCR: ITM Enable bit Position */
#define ITM_TCR_ITMENA_Msk	(1UL /*<< ITM_TCR_ITMENA_Pos*/)	/*!< ITM TCR: ITM Enable bit Mask */

```

/* ITM Integration Write Register Definitions */

#define ITM_IWR_ATVALIDDM_Pos      0           /*!< ITM IWR: ATVALIDDM Position */
/*/

#define ITM_IWR_ATVALIDDM_Msk     (1UL /*<< ITM_IWR_ATVALIDDM_Pos*/)      /*!< ITM IWR:
ATVALIDDM Mask */

/* ITM Integration Read Register Definitions */

#define ITM_IRR_ATREADYM_Pos      0           /*!< ITM IRR: ATREADYM Position */
/*/

#define ITM_IRR_ATREADYM_Msk     (1UL /*<< ITM_IRR_ATREADYM_Pos*/)      /*!< ITM IRR:
ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */

#define ITM_IMCR_INTEGRATION_Pos   0           /*!< ITM IMCR: INTEGRATION
Position */

#define ITM_IMCR_INTEGRATION_Msk   (1UL /*<< ITM_IMCR_INTEGRATION_Pos*/)      /*!< ITM
IMCR: INTEGRATION Mask */

/* ITM Lock Status Register Definitions */

#define ITM_LSR_ByteAcc_Pos        2           /*!< ITM LSR: ByteAcc Position */
#define ITM_LSR_ByteAcc_Msk        (1UL << ITM_LSR_ByteAcc_Pos)      /*!< ITM LSR: ByteAcc
Mask */

#define ITM_LSR_Access_Pos         1           /*!< ITM LSR: Access Position */
#define ITM_LSR_Access_Msk        (1UL << ITM_LSR_Access_Pos)      /*!< ITM LSR: Access
Mask */

#define ITM_LSR_Present_Pos        0           /*!< ITM LSR: Present Position */
#define ITM_LSR_Present_Msk       (1UL /*<< ITM_LSR_Present_Pos*/)      /*!< ITM LSR:
Present Mask */

```

```

/*}@*/ /* end of group CMSIS_ITM */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
\brief Type definitions for the Data Watchpoint and Trace (DWT)

@{
*/

/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).
*/
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) Control Register */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle Count Register */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI Count Register */
    __IO uint32_t EXCCNT;        /*!< Offset: 0x00C (R/W) Exception Overhead Count Register */
    __IO uint32_t SLEEPCNT;      /*!< Offset: 0x010 (R/W) Sleep Count Register */
    __IO uint32_t LSUCNT;        /*!< Offset: 0x014 (R/W) LSU Count Register */
    __IO uint32_t FOLDCNT;       /*!< Offset: 0x018 (R/W) Folded-instruction Count Register */
    __I uint32_t PCSR;          /*!< Offset: 0x01C (R/ ) Program Counter Sample Register */
    __IO uint32_t COMPO;         /*!< Offset: 0x020 (R/W) Comparator Register 0 */
    __IO uint32_t MASK0;         /*!< Offset: 0x024 (R/W) Mask Register 0 */
    __IO uint32_t FUNCTION0;     /*!< Offset: 0x028 (R/W) Function Register 0 */
    uint32_t RESERVED0[1];
    __IO uint32_t COMP1;         /*!< Offset: 0x030 (R/W) Comparator Register 1 */
    __IO uint32_t MASK1;         /*!< Offset: 0x034 (R/W) Mask Register 1 */
    __IO uint32_t FUNCTION1;     /*!< Offset: 0x038 (R/W) Function Register 1 */
    uint32_t RESERVED1[1];
}

```

```

__IO uint32_t COMP2;           /*!< Offset: 0x040 (R/W) Comparator Register 2      */
__IO uint32_t MASK2;          /*!< Offset: 0x044 (R/W) Mask Register 2           */
__IO uint32_t FUNCTION2;       /*!< Offset: 0x048 (R/W) Function Register 2        */
uint32_t RESERVED2[1];

__IO uint32_t COMP3;           /*!< Offset: 0x050 (R/W) Comparator Register 3      */
__IO uint32_t MASK3;          /*!< Offset: 0x054 (R/W) Mask Register 3           */
__IO uint32_t FUNCTION3;       /*!< Offset: 0x058 (R/W) Function Register 3        */
uint32_t RESERVED3[981];

__O uint32_t LAR;             /*!< Offset: 0xFB0 ( W) Lock Access Register       */
__I uint32_t LSR;             /*!< Offset: 0xFB4 ( R ) Lock Status Register      */

} DWT_Type;

/* DWT Control Register Definitions */

#define DWT_CTRL_NUMCOMP_Pos     28                  /*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk     (0xFUL << DWT_CTRL_NUMCOMP_Pos)    /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos    27                  /*!< DWT CTRL: NOTRCPKT Position */
#define DWT_CTRL_NOTRCPKT_Msk    (0x1UL << DWT_CTRL_NOTRCPKT_Pos)   /*!< DWT CTRL: NOTRCPKT Mask */

#define DWT_CTRL_NOEXTTRIG_Pos   26                  /*!< DWT CTRL: NOEXTTRIG Position */
#define DWT_CTRL_NOEXTTRIG_Msk   (0x1UL << DWT_CTRL_NOEXTTRIG_Pos)  /*!< DWT CTRL: NOEXTTRIG Mask */

#define DWT_CTRL_NOCYCCNT_Pos    25                  /*!< DWT CTRL: NOCYCCNT Position */
#define DWT_CTRL_NOCYCCNT_Msk    (0x1UL << DWT_CTRL_NOCYCCNT_Pos)  /*!< DWT CTRL: NOCYCCNT Mask */

```

#define DWT_CTRL_NOPRFCNT_Pos	24	/*!< DWT CTRL: NOPRFCNT Position */
#define DWT_CTRL_NOPRFCNT_Msk	(0x1UL << DWT_CTRL_NOPRFCNT_Pos)	/*!< DWT CTRL: NOPRFCNT Mask */
#define DWT_CTRL_CYCEVTENA_Pos	22	/*!< DWT CTRL: CYCEVTENA Position */
#define DWT_CTRL_CYCEVTENA_Msk	(0x1UL << DWT_CTRL_CYCEVTENA_Pos)	/*!< DWT CTRL: CYCEVTENA Mask */
#define DWT_CTRL_FOLDEVTENA_Pos	21	/*!< DWT CTRL: FOLDEVTENA Position */
#define DWT_CTRL_FOLDEVTENA_Msk	(0x1UL << DWT_CTRL_FOLDEVTENA_Pos)	/*!< DWT CTRL: FOLDEVTENA Mask */
#define DWT_CTRL_LSUEVTENA_Pos	20	/*!< DWT CTRL: LSUEVTENA Position */
#define DWT_CTRL_LSUEVTENA_Msk	(0x1UL << DWT_CTRL_LSUEVTENA_Pos)	/*!< DWT CTRL: LSUEVTENA Mask */
#define DWT_CTRL_SLEEPEVTENA_Pos	19	/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk	(0x1UL << DWT_CTRL_SLEEPEVTENA_Pos)	/*!< DWT CTRL: SLEEPEVTENA Mask */
#define DWT_CTRL_EXCEVTENA_Pos	18	/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk	(0x1UL << DWT_CTRL_EXCEVTENA_Pos)	/*!< DWT CTRL: EXCEVTENA Mask */
#define DWT_CTRL_CPIEVTEA_Pos	17	/*!< DWT CTRL: CPIEVTEA Position */

#define DWT_CTRL_CPIEVTENA_Msk	(0x1UL << DWT_CTRL_CPIEVTENA_Pos)	/*!< DWT CTRL: CPIEVTENA Mask */
#define DWT_CTRL_EXCTRCENA_Pos	16	/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk	(0x1UL << DWT_CTRL_EXCTRCENA_Pos)	/*!< DWT CTRL: EXCTRCENA Mask */
#define DWT_CTRL_PCSAMPLENA_Pos	12	/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk	(0x1UL << DWT_CTRL_PCSAMPLENA_Pos)	/*!< DWT CTRL: PCSAMPLENA Mask */
#define DWT_CTRL_SYNCTAP_Pos	10	/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk	(0x3UL << DWT_CTRL_SYNCTAP_Pos)	/*!< DWT CTRL: SYNCTAP Mask */
#define DWT_CTRL_CYCTAP_Pos	9	/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk	(0x1UL << DWT_CTRL_CYCTAP_Pos)	/*!< DWT CTRL: CYCTAP Mask */
#define DWT_CTRL_POSTINIT_Pos	5	/*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk	(0xFUL << DWT_CTRL_POSTINIT_Pos)	/*!< DWT CTRL: POSTINIT Mask */
#define DWT_CTRL_POSTRESET_Pos	1	/*!< DWT CTRL: POSTRESET Position */
#define DWT_CTRL_POSTRESET_Msk	(0xFUL << DWT_CTRL_POSTRESET_Pos)	/*!< DWT CTRL: POSTRESET Mask */
#define DWT_CTRL_CYCCNTENA_Pos	0	/*!< DWT CTRL: CYCCNTENA Position */

```

#define DWT_CTRL_CYCCNTENA_Msk      (0x1UL /*<< DWT_CTRL_CYCCNTENA_Pos*/) /*!< DWT
CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */

#define DWT_CPICNT_CPICNT_Pos      0                      /*!< DWT CPICNT: CPICNT Position
*/
#define DWT_CPICNT_CPICNT_Msk      (0xFFUL /*<< DWT_CPICNT_CPICNT_Pos*/) /*!< DWT
CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */

#define DWT_EXCCNT_EXCCNT_Pos      0                      /*!< DWT EXCCNT: EXCCNT
Position */
#define DWT_EXCCNT_EXCCNT_Msk      (0xFFUL /*<< DWT_EXCCNT_EXCCNT_Pos*/) /*!< DWT
EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */

#define DWT_SLEEP_CNT_SLEEP_CNT_Pos 0                      /*!< DWT SLEEP_CNT: SLEEP_CNT
Position */
#define DWT_SLEEP_CNT_SLEEP_CNT_Msk (0xFFUL /*<< DWT_SLEEP_CNT_SLEEP_CNT_Pos*/) /*!<
DWT SLEEP_CNT: SLEEP_CNT Mask */

/* DWT LSU Count Register Definitions */

#define DWT_LSUCNT_LSUCNT_Pos      0                      /*!< DWT LSUCNT: LSUCNT Position
*/
#define DWT_LSUCNT_LSUCNT_Msk      (0xFFUL /*<< DWT_LSUCNT_LSUCNT_Pos*/) /*!< DWT
LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */

#define DWT_FOLDCNT_FOLDCNT_Pos    0                      /*!< DWT FOLDCNT: FOLDCNT
Position */
#define DWT_FOLDCNT_FOLDCNT_Msk    (0xFFUL /*<< DWT_FOLDCNT_FOLDCNT_Pos*/) /*!<
DWT FOLDCNT: FOLDCNT Mask */

```

```

/* DWT Comparator Mask Register Definitions */

#define DWT_MASK_MASK_Pos      0           /*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk      (0x1FUL /*<< DWT_MASK_MASK_Pos*/)    /*!< DWT
MASK: MASK Mask */

/* DWT Comparator Function Register Definitions */

#define DWT_FUNCTION_MATCHED_Pos   24          /*!< DWT FUNCTION:
MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk   (0x1UL << DWT_FUNCTION_MATCHED_Pos) /*!<
DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16          /*!< DWT FUNCTION:
DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL << DWT_FUNCTION_DATAVADDR1_Pos) /*!<
DWT FUNCTION: DATAVADDR1 Mask */

#define DWT_FUNCTION_DATAVADDR0_Pos 12          /*!< DWT FUNCTION:
DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk (0xFUL << DWT_FUNCTION_DATAVADDR0_Pos) /*!<
DWT FUNCTION: DATAVADDR0 Mask */

#define DWT_FUNCTION_DATAVSIZE_Pos 10          /*!< DWT FUNCTION:
DATAVSIZE Position */
#define DWT_FUNCTION_DATAVSIZE_Msk (0x3UL << DWT_FUNCTION_DATAVSIZE_Pos) /*!<
DWT FUNCTION: DATAVSIZE Mask */

#define DWT_FUNCTION_LNK1ENA_Pos   9           /*!< DWT FUNCTION: LNK1ENA
Position */
#define DWT_FUNCTION_LNK1ENA_Msk   (0x1UL << DWT_FUNCTION_LNK1ENA_Pos) /*!<
DWT FUNCTION: LNK1ENA Mask */

#define DWT_FUNCTION_DATAVMATCH_Pos 8           /*!< DWT FUNCTION:
DATAVMATCH Position */

```

```

#define DWT_FUNCTION_DATAVMATCH_Msk      (0x1UL << DWT_FUNCTION_DATAVMATCH_Pos)      /*!< DWT FUNCTION: DATAVMATCH Mask */

#define DWT_FUNCTION_CYCMATCH_Pos        7                                         /*!< DWT FUNCTION: CYCMATCH Position */

#define DWT_FUNCTION_CYCMATCH_Msk       (0x1UL << DWT_FUNCTION_CYCMATCH_Pos)      /*!< DWT FUNCTION: CYCMATCH Mask */

#define DWT_FUNCTION_EMITRANGE_Pos      5                                         /*!< DWT FUNCTION: EMITRANGE Position */

#define DWT_FUNCTION_EMITRANGE_Msk     (0x1UL << DWT_FUNCTION_EMITRANGE_Pos)      /*!< DWT FUNCTION: EMITRANGE Mask */

#define DWT_FUNCTION_FUNCTION_Pos       0                                         /*!< DWT FUNCTION: FUNCTION Position */

#define DWT_FUNCTION_FUNCTION_Msk      (0xFUL /*<< DWT_FUNCTION_FUNCTION_Pos*/) /*!< DWT FUNCTION: FUNCTION Mask */

/*}@*/ /* end of group CMSIS_DWT */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_TPI Trace Port Interface (TPI)
\brief Type definitions for the Trace Port Interface (TPI)
@{

```

```

/** \brief Structure type to access the Trace Port Interface Register (TPI).
*/

```

```

typedef struct
{

```

```

__IO uint32_t SSPSR;           /*!< Offset: 0x000 (R/ ) Supported Parallel Port Size Register */
__IO uint32_t CSPSR;           /*!< Offset: 0x004 (R/W) Current Parallel Port Size Register */

    uint32_t RESERVED0[2];

__IO uint32_t ACPR;            /*!< Offset: 0x010 (R/W) Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];

__IO uint32_t SPPR;            /*!< Offset: 0x0F0 (R/W) Selected Pin Protocol Register */
    uint32_t RESERVED2[131];

__I uint32_t FFSR;             /*!< Offset: 0x300 (R/ ) Formatter and Flush Status Register */
__IO uint32_t FFCR;             /*!< Offset: 0x304 (R/W) Formatter and Flush Control Register */
__I uint32_t FSCR;             /*!< Offset: 0x308 (R/ ) Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];

__I uint32_t TRIGGER;          /*!< Offset: 0xEE8 (R/ ) TRIGGER */
__I uint32_t FIFO0;             /*!< Offset: 0xEEC (R/ ) Integration ETM Data */
__I uint32_t ITATBCTR2;         /*!< Offset: 0xEFO (R/ ) ITATBCTR2 */
    uint32_t RESERVED4[1];

__I uint32_t ITATBCTR0;         /*!< Offset: 0xEF8 (R/ ) ITATBCTR0 */
__I uint32_t FIFO1;             /*!< Offset: 0xEFC (R/ ) Integration ITM Data */
__IO uint32_t ITCTRL;            /*!< Offset: 0xF00 (R/W) Integration Mode Control */
    uint32_t RESERVED5[39];

__IO uint32_t CLAIMSET;          /*!< Offset: 0xFA0 (R/W) Claim tag set */
__IO uint32_t CLAIMCLR;          /*!< Offset: 0xFA4 (R/W) Claim tag clear */
    uint32_t RESERVED7[8];

__I uint32_t DEVID;             /*!< Offset: 0xFC8 (R/ ) TPIU_DEVID */
__I uint32_t DEVTYPE;            /*!< Offset: 0xFCC (R/ ) TPIU_DEVTYPE */

} TPI_Type;

```

/* TPI Asynchronous Clock Prescaler Register Definitions */

```
#define TPI_ACPR_PRESCALER_Pos      0                      /*!< TPI ACPR: PRESCALER Position */
*/
```

```

#define TPI_ACPR_PRESCALER_Msk      (0x1FFFUL /*<< TPI_ACPR_PRESCALER_Pos*/) /*!< TPI
ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */

#define TPI_SPPR_TXMODE_Pos        0                      /*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk        (0x3UL /*<< TPI_SPPR_TXMODE_Pos*/) /*!< TPI SPPR:
TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */

#define TPI_FFSR_FtNonStop_Pos    3                      /*!< TPI FFSR: FtNonStop Position */
#define TPI_FFSR_FtNonStop_Msk    (0x1UL << TPI_FFSR_FtNonStop_Pos) /*!< TPI FFSR:
FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos    2                      /*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk    (0x1UL << TPI_FFSR_TCPresent_Pos) /*!< TPI FFSR:
TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos     1                      /*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk    (0x1UL << TPI_FFSR_FtStopped_Pos) /*!< TPI FFSR:
FtStopped Mask */

#define TPI_FFSR_FInProg_Pos       0                      /*!< TPI FFSR: FInProg Position */
#define TPI_FFSR_FInProg_Msk      (0x1UL /*<< TPI_FFSR_FInProg_Pos*/) /*!< TPI FFSR:
FInProg Mask */

/* TPI Formatter and Flush Control Register Definitions */

#define TPI_FFCR_TrigIn_Pos       8                      /*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk       (0x1UL << TPI_FFCR_TrigIn_Pos) /*!< TPI FFCR: TrigIn Mask
*/
#define TPI_FFCR_EnFCont_Pos      1                      /*!< TPI FFCR: EnFCont Position */

```

```

#define TPI_FFCR_EnFCont_Msk      (0x1UL << TPI_FFCR_EnFCont_Pos)      /*!< TPI FFCR:
EnFCont Mask */

/* TPI TRIGGER Register Definitions */

#define TPI_TRIGGER_TRIGGER_Pos    0                                /*!< TPI TRIGGER: TRIGGER Position */
*/

#define TPI_TRIGGER_TRIGGER_Msk    (0x1UL /*<< TPI_TRIGGER_TRIGGER_Pos*/) /*!< TPI
TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */

#define TPI_FIFO0_ITM_ATVALID_Pos  29                             /*!< TPI FIFO0: ITM_ATVALID
Position */

#define TPI_FIFO0_ITM_ATVALID_Msk  (0x3UL << TPI_FIFO0_ITM_ATVALID_Pos) /*!< TPI FIFO0:
ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos 27                            /*!< TPI FIFO0: ITM_bytecount
Position */

#define TPI_FIFO0_ITM_bytecount_Msk (0x3UL << TPI_FIFO0_ITM_bytecount_Pos) /*!< TPI
FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos  26                            /*!< TPI FIFO0: ETM_ATVALID
Position */

#define TPI_FIFO0_ETM_ATVALID_Msk  (0x3UL << TPI_FIFO0_ETM_ATVALID_Pos) /*!< TPI
FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos 24                            /*!< TPI FIFO0: ETM_bytecount
Position */

#define TPI_FIFO0_ETM_bytecount_Msk (0x3UL << TPI_FIFO0_ETM_bytecount_Pos) /*!< TPI
FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos         16                            /*!< TPI FIFO0: ETM2 Position */

#define TPI_FIFO0_ETM2_Msk        (0xFFUL << TPI_FIFO0_ETM2_Pos) /*!< TPI FIFO0: ETM2
Mask */

```

```

#define TPI_FIFO0_ETM1_Pos      8           /*!< TPI FIFO0: ETM1 Position */
#define TPI_FIFO0_ETM1_Msk      (0xFFUL << TPI_FIFO0_ETM1_Pos)    /*!< TPI FIFO0: ETM1
Mask */

#define TPI_FIFO0_ETM0_Pos      0           /*!< TPI FIFO0: ETM0 Position */
#define TPI_FIFO0_ETM0_Msk      (0xFFUL /*<< TPI_FIFO0_ETM0_Pos*/)  /*!< TPI FIFO0:
ETM0 Mask */

/* TPI ITATBCTR2 Register Definitions */

#define TPI_ITATBCTR2_ATREADY_Pos 0           /*!< TPI ITATBCTR2: ATREADY
Position */
#define TPI_ITATBCTR2_ATREADY_Msk (0x1UL /*<< TPI_ITATBCTR2_ATREADY_Pos*/) /*!< TPI
ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */

#define TPI_FIFO1_ITM_ATVALID_Pos 29          /*!< TPI FIFO1: ITM_ATVALID
Position */
#define TPI_FIFO1_ITM_ATVALID_Msk (0x3UL << TPI_FIFO1_ITM_ATVALID_Pos) /*!< TPI FIFO1:
ITM_ATVALID Mask */

#define TPI_FIFO1_ITM_bytecount_Pos 27         /*!< TPI FIFO1: ITM_bytecount
Position */
#define TPI_FIFO1_ITM_bytecount_Msk (0x3UL << TPI_FIFO1_ITM_bytecount_Pos) /*!< TPI
FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos 26          /*!< TPI FIFO1: ETM_ATVALID
Position */
#define TPI_FIFO1_ETM_ATVALID_Msk (0x3UL << TPI_FIFO1_ETM_ATVALID_Pos) /*!< TPI
FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos 24         /*!< TPI FIFO1: ETM_bytecount
Position */

```

```

#define TPI_FIFO1_ETM_bytecount_Msk      (0x3UL << TPI_FIFO1_ETM_bytecount_Pos) /*!< TPI
FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos              16                         /*!< TPI FIFO1: ITM2 Position */
#define TPI_FIFO1_ITM2_Msk              (0xFFUL << TPI_FIFO1_ITM2_Pos) /*!< TPI FIFO1: ITM2
Mask */

#define TPI_FIFO1_ITM1_Pos              8                          /*!< TPI FIFO1: ITM1 Position */
#define TPI_FIFO1_ITM1_Msk              (0xFFUL << TPI_FIFO1_ITM1_Pos) /*!< TPI FIFO1: ITM1
Mask */

#define TPI_FIFO1_ITM0_Pos              0                          /*!< TPI FIFO1: ITM0 Position */
#define TPI_FIFO1_ITM0_Msk              (0xFFUL /*<< TPI_FIFO1_ITM0_Pos*/) /*!< TPI FIFO1: ITM0
Mask */

/* TPI ITATBCTR0 Register Definitions */

#define TPI_ITATBCTR0_ATREADY_Pos       0                          /*!< TPI ITATBCTR0: ATREADY
Position */
#define TPI_ITATBCTR0_ATREADY_Msk       (0x1UL /*<< TPI_ITATBCTR0_ATREADY_Pos*/) /*!< TPI
ITATBCTR0: ATREADY Mask */

/* TPI Integration Mode Control Register Definitions */

#define TPI_ITCTRL_Mode_Pos            0                          /*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk            (0x1UL /*<< TPI_ITCTRL_Mode_Pos*/) /*!< TPI ITCTRL:
Mode Mask */

/* TPI DEVID Register Definitions */

#define TPI_DEVID_NRZVALID_Pos         11                         /*!< TPI DEVID: NRZVALID Position
*/
#define TPI_DEVID_NRZVALID_Msk         (0x1UL << TPI_DEVID_NRZVALID_Pos) /*!< TPI DEVID:
NRZVALID Mask */

```

```

#define TPI_DEVID_MANCVALID_Pos      10          /*!< TPI DEVID: MANCVALID
Position */

#define TPI_DEVID_MANCVALID_Msk     (0x1UL << TPI_DEVID_MANCVALID_Pos) /*!< TPI
DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALIDID_Pos    9           /*!< TPI DEVID: PTINVALIDID Position
*/

#define TPI_DEVID_PTINVALIDID_Msk   (0x1UL << TPI_DEVID_PTINVALIDID_Pos) /*!< TPI DEVID:
PTINVALIDID Mask */

#define TPI_DEVID_MinBufSz_Pos       6           /*!< TPI DEVID: MinBufSz Position */

#define TPI_DEVID_MinBufSz_Msk      (0x7UL << TPI_DEVID_MinBufSz_Pos) /*!< TPI DEVID:
MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos      5           /*!< TPI DEVID: AsynClkIn Position */

#define TPI_DEVID_AsynClkIn_Msk     (0x1UL << TPI_DEVID_AsynClkIn_Pos) /*!< TPI DEVID:
AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos    0           /*!< TPI DEVID: NrTraceInput
Position */

#define TPI_DEVID_NrTraceInput_Msk   (0xFUL /*<< TPI_DEVID_NrTraceInput_Pos*/) /*!< TPI
DEVID: NrTraceInput Mask */

/* TPI DEVTYPE Register Definitions */

#define TPI_DEVTYPE_MajorType_Pos    4           /*!< TPI DEVTYPE: MajorType
Position */

#define TPI_DEVTYPE_MajorType_Msk   (0xFUL << TPI_DEVTYPE_MajorType_Pos) /*!< TPI
DEVTYPE: MajorType Mask */

#define TPI_DEVTYPE_SubType_Pos      0           /*!< TPI DEVTYPE: SubType Position
*/

#define TPI_DEVTYPE_SubType_Msk     (0xFUL /*<< TPI_DEVTYPE_SubType_Pos*/) /*!< TPI
DEVTYPE: SubType Mask */

```

```

/*@}/* /* end of group CMSIS_TPI */

#ifndef __MPU_PRESENT
#define __MPU_PRESENT 0
#endif

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_MPU Memory Protection Unit (MPU)
 \brief Type definitions for the Memory Protection Unit (MPU)

 @{
 */

/** \brief Structure type to access the Memory Protection Unit (MPU).
 */
typedef struct
{
    __I uint32_t TYPE;           /*!< Offset: 0x000 (R/ ) MPU Type Register          */
    __IO uint32_t CTRL;          /*!< Offset: 0x004 (R/W) MPU Control Register        */
    __IO uint32_t RNR;           /*!< Offset: 0x008 (R/W) MPU Region RNRber Register   */
    __IO uint32_t RBAR;          /*!< Offset: 0x00C (R/W) MPU Region Base Address Register */
    __IO uint32_t RASR;          /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register */
    __IO uint32_t RBAR_A1;       /*!< Offset: 0x014 (R/W) MPU Alias 1 Region Base Address Register */
    __IO uint32_t RASR_A1;       /*!< Offset: 0x018 (R/W) MPU Alias 1 Region Attribute and Size Register */
    __IO uint32_t RBAR_A2;       /*!< Offset: 0x01C (R/W) MPU Alias 2 Region Base Address Register */
    __IO uint32_t RASR_A2;       /*!< Offset: 0x020 (R/W) MPU Alias 2 Region Attribute and Size Register */
    __IO uint32_t RBAR_A3;       /*!< Offset: 0x024 (R/W) MPU Alias 3 Region Base Address Register */
}

```

```

__IO uint32_t RASR_A3;           /*!< Offset: 0x028 (R/W) MPU Alias 3 Region Attribute and Size
Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16          /*!< MPU TYPE: IREGION Position
*/
#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)    /*!< MPU TYPE:
IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8           /*!< MPU TYPE: DREGION Position
*/
#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)    /*!< MPU
TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0           /*!< MPU TYPE: SEPARATE
Position */
#define MPU_TYPE_SEPARATE_Msk     (1UL /*<< MPU_TYPE_SEPARATE_Pos*/)  /*!< MPU
TYPE: SEPARATE Mask */

/* MPU Control Register */

#define MPU_CTRL_PRIVDEFENA_Pos   2           /*!< MPU CTRL: PRIVDEFENA
Position */
#define MPU_CTRL_PRIVDEFENA_Msk   (1UL << MPU_CTRL_PRIVDEFENA_Pos)    /*!< MPU
CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos    1           /*!< MPU CTRL: HFNMIENA
Position */
#define MPU_CTRL_HFNMIENA_Msk    (1UL << MPU_CTRL_HFNMIENA_Pos)    /*!< MPU
CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos       0           /*!< MPU CTRL: ENABLE Position */

```

```

#define MPU_CTRL_ENABLE_Msk      (1UL /*<< MPU_CTRL_ENABLE_Pos*/)      /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */

#define MPU_RNR_REGION_Pos      0                                /*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk      (0xFFUL /*<< MPU_RNR_REGION_Pos*/)      /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos       5                                /*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk       (0x7FFFFFFUL << MPU_RBAR_ADDR_Pos)      /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos      4                                /*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk      (1UL << MPU_RBAR_VALID_Pos)      /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos     0                                /*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk     (0xFUL /*<< MPU_RBAR_REGION_Pos*/)      /*!< MPU
RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */

#define MPU_RASR_ATTRS_Pos      16                             /*!< MPU RASR: MPU Region
Attribute field Position */
#define MPU_RASR_ATTRS_Msk      (0xFFFFUL << MPU_RASR_ATTRS_Pos)      /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28                             /*!< MPU RASR: ATTRS.XN Position */
#define MPU_RASR_XN_Msk         (1UL << MPU_RASR_XN_Pos)      /*!< MPU RASR:
ATTRS.XN Mask */

```

#define MPU_RASR_AP_Pos	24	/*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk	(0x7UL << MPU_RASR_AP_Pos)	/*!< MPU RASR: ATTRS.AP Mask */
#define MPU_RASR_TEX_Pos	19	/*!< MPU RASR: ATTRS.TEX Position */
#define MPU_RASR_TEX_Msk	(0x7UL << MPU_RASR_TEX_Pos)	/*!< MPU RASR: ATTRS.TEX Mask */
#define MPU_RASR_S_Pos	18	/*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk	(1UL << MPU_RASR_S_Pos)	/*!< MPU RASR: ATTRS.S Mask */
#define MPU_RASR_C_Pos	17	/*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk	(1UL << MPU_RASR_C_Pos)	/*!< MPU RASR: ATTRS.C Mask */
#define MPU_RASR_B_Pos	16	/*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk	(1UL << MPU_RASR_B_Pos)	/*!< MPU RASR: ATTRS.B Mask */
#define MPU_RASR_SRD_Pos	8	/*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk	(0xFFUL << MPU_RASR_SRD_Pos)	/*!< MPU RASR: Sub-Region Disable Mask */
#define MPU_RASR_SIZE_Pos	1	/*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk	(0x1FUL << MPU_RASR_SIZE_Pos)	/*!< MPU RASR: Region Size Field Mask */

```
#define MPU_RASR_ENABLE_Pos      0           /*!< MPU RASR: Region enable bit  
Position */  
  
#define MPU_RASR_ENABLE_Msk      (1UL /*<< MPU_RASR_ENABLE_Pos*/)      /*!< MPU  
RASR: Region enable bit Disable Mask */
```

```
/*@} end of group CMSIS_MPU */
```

```
#endif
```

```
#if (__FPU_PRESENT == 1)  
/** \ingroup CMSIS_core_register  
\\defgroup CMSIS_FPU Floating Point Unit (FPU)  
\\brief Type definitions for the Floating Point Unit (FPU)  
{@  
*/
```

```
/** \\brief Structure type to access the Floating Point Unit (FPU).
```

```
*/
```

```
typedef struct
```

```
{
```

```
    uint32_t RESERVED0[1];  
  
    __IO uint32_t FPCCR;          /*!< Offset: 0x004 (R/W) Floating-Point Context Control Register  
*/  
  
    __IO uint32_t FPCAR;          /*!< Offset: 0x008 (R/W) Floating-Point Context Address Register  
*/  
  
    __IO uint32_t FPDSCR;         /*!< Offset: 0x00C (R/W) Floating-Point Default Status Control  
Register */  
  
    __I uint32_t MVFR0;           /*!< Offset: 0x010 (R/ ) Media and FP Feature Register 0 */  
    __I uint32_t MVFR1;           /*!< Offset: 0x014 (R/ ) Media and FP Feature Register 1 */  
    __I uint32_t MVFR2;           /*!< Offset: 0x018 (R/ ) Media and FP Feature Register 2 */
```

```

} FPU_Type;

/* Floating-Point Context Control Register */

#define FPU_FPCCR_AS PEN_Pos      31           /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk       (1UL << FPU_FPCCR_AS PEN_Pos)      /*!< FPCCR: ASPEN
bit Mask */

#define FPU_FPCCR_LSPEN_Pos      30           /*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSPEN_Msk       (1UL << FPU_FPCCR_LSPEN_Pos)      /*!< FPCCR: LSPEN
bit Mask */

#define FPU_FPCCR_MONRDY_Pos     8            /*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MONRDY_Msk      (1UL << FPU_FPCCR_MONRDY_Pos)      /*!< FPCCR:
MONRDY bit Mask */

#define FPU_FPCCR_BFRDY_Pos      6            /*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFRDY_Msk       (1UL << FPU_FPCCR_BFRDY_Pos)      /*!< FPCCR: BFRDY
bit Mask */

#define FPU_FPCCR_MMRDY_Pos      5            /*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMRDY_Msk       (1UL << FPU_FPCCR_MMRDY_Pos)      /*!< FPCCR:
MMRDY bit Mask */

#define FPU_FPCCR_HFRDY_Pos      4            /*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk       (1UL << FPU_FPCCR_HFRDY_Pos)      /*!< FPCCR:
HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos      3           /*!< FPCCR: processor mode bit
Position */
#define FPU_FPCCR_THREAD_Msk      (1UL << FPU_FPCCR_THREAD_Pos)      /*!< FPCCR:
processor mode active bit Mask */

```

```

#define FPU_FPCCR_USER_Pos           1           /*!< FPCCR: privilege level bit
Position */

#define FPU_FPCCR_USER_Msk          (1UL << FPU_FPCCR_USER_Pos)      /*!< FPCCR:
privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos        0           /*!< FPCCR: Lazy state preservation
active bit Position */

#define FPU_FPCCR_LSPACT_Msk        (1UL /*<< FPU_FPCCR_LSPACT_Pos*/)    /*!< FPCCR: Lazy
state preservation active bit Mask */

/* Floating-Point Context Address Register */

#define FPU_FPCAR_ADDRESS_Pos       3           /*!< FPCAR: ADDRESS bit Position
*/
#define FPU_FPCAR_ADDRESS_Msk       (0x1FFFFFFFUL << FPU_FPCAR_ADDRESS_Pos) /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */

#define FPU_FPDSCR_AHP_Pos         26          /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk          (1UL << FPU_FPDSCR_AHP_Pos)      /*!< FPDSCR: AHP
bit Mask */

#define FPU_FPDSCR_DN_Pos          25          /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk          (1UL << FPU_FPDSCR_DN_Pos)      /*!< FPDSCR: DN bit
Mask */

#define FPU_FPDSCR_FZ_Pos          24          /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk          (1UL << FPU_FPDSCR_FZ_Pos)      /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos        22          /*!< FPDSCR: RMode bit Position
*/
#define FPU_FPDSCR_RMode_Msk        (3UL << FPU_FPDSCR_RMode_Pos)    /*!< FPDSCR:
RMode bit Mask */

```

```

/* Media and FP Feature Register 0 */

#define FPU_MVFR0_FP_rounding_modes_Pos 28          /*!< MVFR0: FP rounding
modes bits Position */

#define FPU_MVFR0_FP_rounding_modes_Msk (0xFUL << FPU_MVFR0_FP_rounding_modes_Pos)
/*!< MVFR0: FP rounding modes bits Mask */

#define FPU_MVFR0_Short_vectors_Pos    24          /*!< MVFR0: Short vectors bits
Position */

#define FPU_MVFR0_Short_vectors_Msk   (0xFUL << FPU_MVFR0_Short_vectors_Pos)  /*!<
MVFR0: Short vectors bits Mask */

#define FPU_MVFR0_Square_root_Pos     20          /*!< MVFR0: Square root bits
Position */

#define FPU_MVFR0_Square_root_Msk    (0xFUL << FPU_MVFR0_Square_root_Pos)  /*!<
MVFR0: Square root bits Mask */

#define FPU_MVFR0_Divide_Pos        16          /*!< MVFR0: Divide bits Position */

#define FPU_MVFR0_Divide_Msk        (0xFUL << FPU_MVFR0_Divide_Pos)  /*!< MVFR0:
Divide bits Mask */

#define FPU_MVFR0_FP_excep_trapping_Pos 12          /*!< MVFR0: FP exception
trapping bits Position */

#define FPU_MVFR0_FP_excep_trapping_Msk (0xFUL << FPU_MVFR0_FP_excep_trapping_Pos) /*!<
MVFR0: FP exception trapping bits Mask */

#define FPU_MVFR0_Double_precision_Pos 8           /*!< MVFR0: Double-precision
bits Position */

#define FPU_MVFR0_Double_precision_Msk (0xFUL << FPU_MVFR0_Double_precision_Pos) /*!<
MVFR0: Double-precision bits Mask */

#define FPU_MVFR0_Single_precision_Pos 4           /*!< MVFR0: Single-precision
bits Position */

```

```

#define FPU_MVFR0_Single_precision_Msk  (0xFUL << FPU_MVFR0_Single_precision_Pos) /*!<
MVFR0: Single-precision bits Mask */

#define FPU_MVFR0_A SIMD_registers_Pos    0                                /*!< MVFR0: A SIMD registers
bits Position */

#define FPU_MVFR0_A SIMD_registers_Msk  (0xFUL /*<< FPU_MVFR0_A SIMD_registers_Pos*/) /*!<
MVFR0: A SIMD registers bits Mask */

/* Media and FP Feature Register 1 */

#define FPU_MVFR1_FP_fused_MAC_Pos     28                               /*!< MVFR1: FP fused MAC
bits Position */

#define FPU_MVFR1_FP_fused_MAC_Msk   (0xFUL << FPU_MVFR1_FP_fused_MAC_Pos) /*!<
MVFR1: FP fused MAC bits Mask */

#define FPU_MVFR1_FP_HPFP_Pos        24                               /*!< MVFR1: FP HPFP bits
Position */

#define FPU_MVFR1_FP_HPFP_Msk      (0xFUL << FPU_MVFR1_FP_HPFP_Pos) /*!< MVFR1:
FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos     4                                /*!< MVFR1: D_NaN mode bits
Position */

#define FPU_MVFR1_D_NaN_mode_Msk   (0xFUL << FPU_MVFR1_D_NaN_mode_Pos) /*!<
MVFR1: D_NaN mode bits Mask */

#define FPU_MVFR1_FtZ_mode_Pos       0                               /*!< MVFR1: FtZ mode bits
Position */

#define FPU_MVFR1_FtZ_mode_Msk     (0xFUL /*<< FPU_MVFR1_FtZ_mode_Pos*/) /*!<
MVFR1: FtZ mode bits Mask */

/* Media and FP Feature Register 2 */

/*@} end of group CMSIS_FPU */
#endif

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CoreDebug    Core Debug Registers (CoreDebug)
 \brief    Type definitions for the Core Debug Registers

 @{

 */

/** \brief Structure type to access the Core Debug Register (CoreDebug).

 */
typedef struct
{
    __IO uint32_t DHCSR;           /*!< Offset: 0x000 (R/W) Debug Halting Control and Status Register */
    __O  uint32_t DCRSR;          /*!< Offset: 0x004 ( /W) Debug Core Register Selector Register      */
    __IO uint32_t DCRDR;          /*!< Offset: 0x008 (R/W) Debug Core Register Data Register        */
    __IO uint32_t DEMCR;          /*!< Offset: 0x00C (R/W) Debug Exception and Monitor Control
Register */

} CoreDebug_Type;

/* Debug Halting Control and Status Register */

#define CoreDebug_DHCSR_DBGKEY_Pos     16           /*!< CoreDebug DHCSR:
DBGKEY Position */

#define CoreDebug_DHCSR_DBGKEY_Msk     (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos)  /*!<
CoreDebug DHCSR: DBGKEY Mask */

#define CoreDebug_DHCSR_S_RESET_ST_Pos  25           /*!< CoreDebug DHCSR:
S_RESET_ST Position */

#define CoreDebug_DHCSR_S_RESET_ST_Msk (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos)  /*!<
CoreDebug DHCSR: S_RESET_ST Mask */

```

```

#define CoreDebug_DHCSR_S_RETIRE_ST_Pos  24          /*!< CoreDebug DHCSR:  

S_RETIRE_ST Position */  
  

#define CoreDebug_DHCSR_S_RETIRE_ST_Msk  (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)  /*!<  
CoreDebug DHCSR: S_RETIRE_ST Mask */  
  

#define CoreDebug_DHCSR_S_LOCKUP_Pos    19          /*!< CoreDebug DHCSR:  

S_LOCKUP Position */  
  

#define CoreDebug_DHCSR_S_LOCKUP_Msk   (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)  /*!<  
CoreDebug DHCSR: S_LOCKUP Mask */  
  

#define CoreDebug_DHCSR_S_SLEEP_Pos    18          /*!< CoreDebug DHCSR:  

S_SLEEP Position */  
  

#define CoreDebug_DHCSR_S_SLEEP_Msk   (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)  /*!<  
CoreDebug DHCSR: S_SLEEP Mask */  
  

#define CoreDebug_DHCSR_S_HALT_Pos     17          /*!< CoreDebug DHCSR:  

S_HALT Position */  
  

#define CoreDebug_DHCSR_S_HALT_Msk   (1UL << CoreDebug_DHCSR_S_HALT_Pos)  /*!<  
CoreDebug DHCSR: S_HALT Mask */  
  

#define CoreDebug_DHCSR_S_REGRDY_Pos   16          /*!< CoreDebug DHCSR:  

S_REGRDY Position */  
  

#define CoreDebug_DHCSR_S_REGRDY_Msk  (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)  /*!<  
CoreDebug DHCSR: S_REGRDY Mask */  
  

#define CoreDebug_DHCSR_C_SNAPSTALL_Pos 5          /*!< CoreDebug DHCSR:  

C_SNAPSTALL Position */  
  

#define CoreDebug_DHCSR_C_SNAPSTALL_Msk (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)  
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */  
  

#define CoreDebug_DHCSR_C_MASKINTS_Pos  3          /*!< CoreDebug DHCSR:  

C_MASKINTS Position */  
  

#define CoreDebug_DHCSR_C_MASKINTS_Msk (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)  
/*!< CoreDebug DHCSR: C_MASKINTS Mask */

```

```

#define CoreDebug_DHCSR_C_STEP_Pos      2                  /*!< CoreDebug DHCSR: C_STEP
Position */

#define CoreDebug_DHCSR_C_STEP_Msk      (1UL << CoreDebug_DHCSR_C_STEP_Pos)      /*!<
CoreDebug DHCSR: C_STEP Mask */

#define CoreDebug_DHCSR_C_HALT_Pos     1                  /*!< CoreDebug DHCSR:
C_HALT Position */

#define CoreDebug_DHCSR_C_HALT_Msk     (1UL << CoreDebug_DHCSR_C_HALT_Pos)      /*!<
CoreDebug DHCSR: C_HALT Mask */

#define CoreDebug_DHCSR_C_DEBUGEN_Pos   0                  /*!< CoreDebug DHCSR:
C_DEBUGEN Position */

#define CoreDebug_DHCSR_C_DEBUGEN_Msk   (1UL /*<< CoreDebug_DHCSR_C_DEBUGEN_Pos*/) /*!<
CoreDebug DHCSR: C_DEBUGEN Mask */

/* Debug Core Register Selector Register */

#define CoreDebug_DCRSR_REGWnR_Pos     16                 /*!< CoreDebug DCRSR:
REGWnR Position */

#define CoreDebug_DCRSR_REGWnR_Msk     (1UL << CoreDebug_DCRSR_REGWnR_Pos)      /*!<
CoreDebug DCRSR: REGWnR Mask */

#define CoreDebug_DCRSR_REGSEL_Pos     0                  /*!< CoreDebug DCRSR: REGSEL
Position */

#define CoreDebug_DCRSR_REGSEL_Msk     (0x1FUL /*<< CoreDebug_DCRSR_REGSEL_Pos*/) /*!<
CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos    24                 /*!< CoreDebug DEMCR:
TRCENA Position */

#define CoreDebug_DEMCR_TRCENA_Msk    (1UL << CoreDebug_DEMCR_TRCENA_Pos)      /*!<
CoreDebug DEMCR: TRCENA Mask */

```

```

#define CoreDebug_DEMCR_MON_REQ_Pos    19           /*!< CoreDebug DEMCR:  

MON_REQ Position */  
  

#define CoreDebug_DEMCR_MON_REQ_Msk   (1UL << CoreDebug_DEMCR_MON_REQ_Pos)  

/*!< CoreDebug DEMCR: MON_REQ Mask */  
  

#define CoreDebug_DEMCR_MON_STEP_Pos  18           /*!< CoreDebug DEMCR:  

MON_STEP Position */  
  

#define CoreDebug_DEMCR_MON_STEP_Msk  (1UL << CoreDebug_DEMCR_MON_STEP_Pos)  

/*!< CoreDebug DEMCR: MON_STEP Mask */  
  

#define CoreDebug_DEMCR_MON_PEND_Pos  17           /*!< CoreDebug DEMCR:  

MON_PEND Position */  
  

#define CoreDebug_DEMCR_MON_PEND_Msk  (1UL << CoreDebug_DEMCR_MON_PEND_Pos)  

/*!< CoreDebug DEMCR: MON_PEND Mask */  
  

#define CoreDebug_DEMCR_MON_EN_Pos    16           /*!< CoreDebug DEMCR:  

MON_EN Position */  
  

#define CoreDebug_DEMCR_MON_EN_Msk    (1UL << CoreDebug_DEMCR_MON_EN_Pos)      /*!<  

CoreDebug DEMCR: MON_EN Mask */  
  

#define CoreDebug_DEMCR_VC_HARDERR_Pos 10          /*!< CoreDebug DEMCR:  

VC_HARDERR Position */  
  

#define CoreDebug_DEMCR_VC_HARDERR_Msk (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos)  

/*!< CoreDebug DEMCR: VC_HARDERR Mask */  
  

#define CoreDebug_DEMCR_VC_INTERR_Pos  9            /*!< CoreDebug DEMCR:  

VC_INTERR Position */  
  

#define CoreDebug_DEMCR_VC_INTERR_Msk  (1UL << CoreDebug_DEMCR_VC_INTERR_Pos)    /*!<  

CoreDebug DEMCR: VC_INTERR Mask */  
  

#define CoreDebug_DEMCR_VC_BUSERR_Pos  8            /*!< CoreDebug DEMCR:  

VC_BUSERR Position */  
  

#define CoreDebug_DEMCR_VC_BUSERR_Msk  (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos)  

/*!< CoreDebug DEMCR: VC_BUSERR Mask */

```

```

#define CoreDebug_DEMCR_VC_STATERR_Pos    7           /*!< CoreDebug DEMCR:  

VC_STATERR Position */  
  

#define CoreDebug_DEMCR_VC_STATERR_Msk   (1UL << CoreDebug_DEMCR_VC_STATERR_Pos)  

/*!< CoreDebug DEMCR: VC_STATERR Mask */  
  

#define CoreDebug_DEMCR_VC_CHKERR_Pos    6           /*!< CoreDebug DEMCR:  

VC_CHKERR Position */  
  

#define CoreDebug_DEMCR_VC_CHKERR_Msk   (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos)  

/*!< CoreDebug DEMCR: VC_CHKERR Mask */  
  

#define CoreDebug_DEMCR_VC_NOCPERR_Pos   5           /*!< CoreDebug DEMCR:  

VC_NOCPERR Position */  
  

#define CoreDebug_DEMCR_VC_NOCPERR_Msk  (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos)  

/*!< CoreDebug DEMCR: VC_NOCPERR Mask */  
  

#define CoreDebug_DEMCR_VC_MMERR_Pos    4           /*!< CoreDebug DEMCR:  

VC_MMERR Position */  
  

#define CoreDebug_DEMCR_VC_MMERR_Msk   (1UL << CoreDebug_DEMCR_VC_MMERR_Pos)  

/*!< CoreDebug DEMCR: VC_MMERR Mask */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Pos 0           /*!< CoreDebug DEMCR:  

VC_CORERESET Position */  
  

#define CoreDebug_DEMCR_VC_CORERESET_Msk (1UL /*<<  

CoreDebug_DEMCR_VC_CORERESET_Pos*/) /*!< CoreDebug DEMCR: VC_CORERESET Mask */

```

/*@} end of group CMSIS_CoreDebug */

```

/** \ingroup CMSIS_core_register  

\defgroup CMSIS_core_base Core Definitions  

\brief Definitions for base addresses, unions, and structures.

```

```

@{

/* Memory mapping of Cortex-M4 Hardware */

#define SCS_BASE      (0xE000E000UL)           /*!< System Control Space Base Address */
#define ITM_BASE      (0xE0000000UL)           /*!< ITM Base Address */
#define DWT_BASE      (0xE0001000UL)           /*!< DWT Base Address */
#define TPI_BASE      (0xE0040000UL)           /*!< TPI Base Address */
#define CoreDebug_BASE (0xE000EDFOUL)          /*!< Core Debug Base Address */
#define SysTick_BASE   (SCS_BASE + 0x0010UL)       /*!< SysTick Base Address */
#define NVIC_BASE     (SCS_BASE + 0x0100UL)          /*!< NVIC Base Address */
#define SCB_BASE       (SCS_BASE + 0x0D00UL)          /*!< System Control Block Base Address */

#define SCnSCB        ((SCnSCB_Type *) SCS_BASE) /*!< System control Register not in SCB */
#define SCB           ((SCB_Type *) SCB_BASE)        /*!< SCB configuration struct */
#define SysTick       ((SysTick_Type *) SysTick_BASE) /*!< SysTick configuration struct */
#define NVIC          ((NVIC_Type *) NVIC_BASE)        /*!< NVIC configuration struct */
#define ITM           ((ITM_Type *) ITM_BASE)         /*!< ITM configuration struct */
#define DWT           ((DWT_Type *) DWT_BASE)         /*!< DWT configuration struct */
#define TPI           ((TPI_Type *) TPI_BASE)         /*!< TPI configuration struct */
#define CoreDebug     ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug configuration
struct */

#if (__MPU_PRESENT == 1)

#define MPU_BASE      (SCS_BASE + 0x0D90UL)          /*!< Memory Protection Unit */
#define MPU          ((MPU_Type *) MPU_BASE)        /*!< Memory Protection Unit */

#endif

#if (__FPU_PRESENT == 1)

```

```
#define FPU_BASE      (SCS_BASE + 0x0F30UL)           /*!< Floating Point Unit      */
#define FPU      ((FPU_Type    *)  FPU_BASE   ) /*!< Floating Point Unit      */
#endif

/*@} */
```

```
*****
```

```
*       Hardware Abstraction Layer
```

```
Core Function Interface contains:
```

- Core NVIC Functions
- Core SysTick Functions
- Core Debug Functions
- Core Register Access Functions

```
*****/
```

```
/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/
```

```
/* ##### NVIC functions ##### */
```

```
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the NVIC.
@{
*/
```

```
/** \brief Set Priority Grouping
```

The function sets the priority grouping field using the required unlock sequence.

The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.

Only values from 0..7 are used.

In case of a conflict between priority grouping and available

priority bits (`__NVIC_PRIO_BITS`), the smallest possible priority group is set.

```
\param [in] PriorityGroup Priority grouping field.  
*/  
__STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)  
{  
    uint32_t reg_value;  
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL);      /* only values 0..7 are used  
 */  
  
    reg_value = SCB->AIRCR;                      /* read old register configuration */  
    reg_value &= ~((uint32_t)(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk));      /* clear  
 bits to change */  
    reg_value = (reg_value  
                |  
                ((uint32_t)0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |  
                (PriorityGroupTmp << 8));      /* Insert write key and priority group */  
    SCB->AIRCR = reg_value;  
}  
  
/** \brief Get Priority Grouping
```

The function reads the priority grouping field from the NVIC Interrupt Controller.

```
\return Priority grouping field (SCB->AIRCR [10:8] PRIGROUP field).  
*/  
STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)  
{  
    return ((uint32_t)((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos));  
}
```

```
/** \brief Enable External Interrupt
```

The function enables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
    0x1FUL));  
}
```

```
/** \brief Disable External Interrupt
```

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in] IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)  
{
```

```
NVIC->ICER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
0x1FUL));  
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not pending.

\return 1 Interrupt status is pending.

```
*/
```

```
__STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)  
{  
    return((uint32_t)((NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL]) & (1UL << (((uint32_t)(int32_t)IRQn)  
& 0x1FUL))) != 0UL) ? 1UL : 0UL);  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

\param [in] IRQn Interrupt number. Value cannot be negative.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
```

```
{  
    NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
    0x1FUL));  
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```
\param [in]    IRQn External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &  
    0x1FUL));  
}
```

```
/** \brief Get Active Interrupt
```

The function reads the active register in NVIC and returns the active bit.

```
\param [in]    IRQn Interrupt number.  
  
\return      0 Interrupt status is not active.  
\return      1 Interrupt status is active.  
*/  
STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
```

```
{  
    return((uint32_t)((NVIC->IABR[((uint32_t)(int32_t)IRQn) >> 5UL] & (1UL << (((uint32_t)(int32_t)IRQn)  
    & 0x1FUL)) != 0UL) ? 1UL : 0UL);  
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.

\param [in] priority Priority to set.

```
*/
```

```
__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)  
{  
    if((int32_t)IRQn < 0) {  
        SCB->SHPR[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS))  
        & (uint32_t)0xFFUL);  
    }  
    else {  
        NVIC->IP[((uint32_t)(int32_t)IRQn)] = (uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) &  
        (uint32_t)0xFFUL);  
    }  
}
```

```
/** \brief Get Interrupt Priority
```

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

```
\param [in] IRQn Interrupt number.  
\return Interrupt Priority. Value is aligned automatically to the implemented  
priority bits of the microcontroller.  
*/  
STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)  
{  
  
if((int32_t)IRQn < 0) {  
    return(((uint32_t)SCB->SHPR[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] >> (8 - __NVIC_PRIO_BITS));  
}  
else {  
    return(((uint32_t)NVIC->IP[((uint32_t)(int32_t)IRQn)] >> (8 - __NVIC_PRIO_BITS));  
}  
}  
  
/** \brief Encode Priority
```

The function encodes the priority for an interrupt with the given priority group, preemptive priority value, and subpriority value.

In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS), the smallest possible priority group is set.

```
\param [in] PriorityGroup Used priority group.
```

```

\param [in] PreemptPriority Preemptive priority value (starting from 0).
\param [in] SubPriority Subpriority value (starting from 0).
\return Encoded priority. Value can be used in the function \ref NVIC_SetPriority().

*/
__STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority,
                                            uint32_t SubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
        (uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);

    SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
        (uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));

    return (
        ((PreemptPriority & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL)) << SubPriorityBits) |
        ((SubPriority & (uint32_t)((1UL << (SubPriorityBits)) - 1UL)))
    );
}

```

/** \brief Decode Priority

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.
In case of a conflict between priority grouping and available priority bits (NVIC_PRIO_BITS) the smallest possible priority group is set.

```

    \param [in]      Priority  Priority value, which can be retrieved with the function \ref
NVIC_GetPriority().

    \param [in]  PriorityGroup  Used priority group.

    \param [out] pPreemptPriority  Preemptive priority value (starting from 0).

    \param [out] pSubPriority  Subpriority value (starting from 0).

*/
__STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t*
pPreemptPriority, uint32_t* pSubPriority)

{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /* only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) > (uint32_t)(__NVIC_PRIO_BITS)) ?
(uint32_t)(__NVIC_PRIO_BITS) : (uint32_t)(7UL - PriorityGroupTmp);

    SubPriorityBits = ((PriorityGroupTmp + (uint32_t)(__NVIC_PRIO_BITS)) < (uint32_t)7UL) ?
(uint32_t)0UL : (uint32_t)((PriorityGroupTmp - 7UL) + (uint32_t)(__NVIC_PRIO_BITS));

    *pPreemptPriority = (Priority >> SubPriorityBits) & (uint32_t)((1UL << (PreemptPriorityBits)) - 1UL);
    *pSubPriority = (Priority ) & (uint32_t)((1UL << (SubPriorityBits )) - 1UL);
}

/** \brief System Reset

```

The function initiates a system reset request to reset the MCU.

```

*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB(); /* Ensure all outstanding memory accesses included

```

```

        buffered write are completed before reset */

SCB->AIRCR = (uint32_t)((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
    (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
    SCB_AIRCR_SYSRESETREQ_Msk ); /* Keep priority group unchanged */

__DSB(); /* Ensure completion of memory access */

while(1) { __NOP(); } /* wait until reset */

}

```

/*@} end of CMSIS_Core_NVICFunctions */

```

/* ##### FPU functions #####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_FpuFunctions FPU Functions
 \brief Function that provides FPU type.

 @{
 */

/**
\fn     uint32_t SCB_GetFPType(void)
\brief   get FPU type
\returns
- \b 0: No FPU
- \b 1: Single precision FPU
- \b 2: Double + Single precision FPU
*/
__STATIC_INLINE uint32_t SCB_GetFPType(void)
{
    uint32_t mvfr0;

```

```

mvfr0 = SCB->MVFR0;

if      ((mvfr0 & 0x00000FF0UL) == 0x220UL) {
    return 2UL;      // Double + Single precision FPU
} else if ((mvfr0 & 0x00000FF0UL) == 0x020UL) {
    return 1UL;      // Single precision FPU
} else {
    return 0UL;      // No FPU
}
}

```

```
/*@} end of CMSIS_Core_FpuFunctions */
```

```
/* ##### Cache functions ##### */
```

```

/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_CacheFunctions Cache Functions
\brief Functions that configure Instruction and Data cache.

@{
*/

```

```
/* Cache Size ID Register Macros */
```

```

#define CCSIDR_WAYS(x)    (((x) & SCB_CCSIDR_ASSOCIATIVITY_Msk) >>
SCB_CCSIDR_ASSOCIATIVITY_Pos)

#define CCSIDR_SETS(x)     (((x) & SCB_CCSIDR_NUMSETS_Msk) >> SCB_CCSIDR_NUMSETS_Pos)

#define CCSIDR_LSSHIFT(x)   (((x) & SCB_CCSIDR_LINESIZE_Msk) /*>> SCB_CCSIDR_LINESIZE_Pos*/ )

```

```
/** \brief Enable I-Cache

The function turns on I-Cache

*/
__STATIC_INLINE void SCB_EnableICache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->ICIALLU = 0UL;           // invalidate I-Cache
    SCB->CCR |= (uint32_t)SCB_CCR_IC_Msk; // enable I-Cache
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Disable I-Cache

The function turns off I-Cache

*/
__STATIC_INLINE void SCB_DisableICache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->CCR &= ~(uint32_t)SCB_CCR_IC_Msk; // disable I-Cache
    SCB->ICIALLU = 0UL;           // invalidate I-Cache
}
```

```
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Invalidate I-Cache
```

The function invalidates I-Cache

```
 */
__STATIC_INLINE void SCB_InvalidateCache (void)
{
#if (__ICACHE_PRESENT == 1)
    __DSB();
    __ISB();
    SCB->ICIALLU = 0UL;
    __DSB();
    __ISB();
#endif
}
```

```
/** \brief Enable D-Cache
```

The function turns on D-Cache

```
 */
__STATIC_INLINE void SCB_EnableDCache (void)
{
#if (__DCACHE_PRESENT == 1)
```

```

uint32_t ccsidr, sshift, wshift, sw;

uint32_t sets, ways;

SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
ccsidr = SCB->CCSIDR;
sets = (uint32_t)(CCSIDR_SETS(ccsidr));
sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

__DSB();

do {                                // invalidate D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCISW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();

SCB->CCR |= (uint32_t)SCB_CCR_DC_Msk; // enable D-Cache

__DSB();
__ISB();
#endif
}

```

```

/** \brief Disable D-Cache

The function turns off D-Cache

*/
__STATIC_INLINE void SCB_DisableDCache (void)
{
#if (__DCACHE_PRESENT == 1)

    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

    __DSB();

    SCB->CCR &= ~(uint32_t)SCB_CCR_DC_Msk; // disable D-Cache

    do {                                // clean & invalidate D-Cache
        uint32_t tmpways = ways;
        do {
            sw = ((tmpways << wshift) | (sets << sshift));
            SCB->DCCISW = sw;
        } while(tmpways--);
    } while(sets--);
}

```

```
    __DSB();  
    __ISB();  
#endif  
}  
  
/*\brief Invalidate D-Cache
```

```
The function invalidates D-Cache  
*/  
__STATIC_INLINE void SCB_InvalidateDCache (void)  
{  
#if (__DCACHE_PRESENT == 1)  
    uint32_t ccsidr, sshift, wshift, sw;  
    uint32_t sets, ways;  
  
    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache  
    ccsidr = SCB->CCSIDR;  
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));  
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);  
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));  
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);  
  
    __DSB();  
  
    do {                      // invalidate D-Cache  
        uint32_t tmpways = ways;  
        do {
```

```

    sw = ((tmpways << wshift) | (sets << sshift));
    SCB->DCISW = sw;
} while(tmpways--);

} while(sets--);

__DSB();
__ISB();

#endif
}

/** \brief Clean D-Cache

The function cleans D-Cache

*/
__STATIC_INLINE void SCB_CleanDCache (void)
{
#ifndef __DCACHE_PRESENT == 1
    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (OUL << 1) | OUL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);
    ways = (uint32_t)(CCSIDR_WAYS(ccsidr));
    wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

    __DSB();

```

```

do {                                // clean D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCCSW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();
__ISB();
#endif
}

/** \brief Clean & Invalidate D-Cache

The function cleans and Invalidates D-Cache
*/
__STATIC_INLINE void SCB_CleanInvalidateDCache (void)
{
#if (__DCACHE_PRESENT == 1)
    uint32_t ccsidr, sshift, wshift, sw;
    uint32_t sets, ways;

    SCB->CSSELR = (0UL << 1) | 0UL;      // Level 1 data cache
    ccsidr = SCB->CCSIDR;
    sets = (uint32_t)(CCSIDR_SETS(ccsidr));
    sshift = (uint32_t)(CCSIDR_LSSHIFT(ccsidr) + 4UL);

```

```

ways  = (uint32_t)(CCSIDR_WAYS(ccsidr));
wshift = (uint32_t)((uint32_t)__CLZ(ways) & 0x1FUL);

__DSB();

do {           // clean & invalidate D-Cache
    uint32_t tmpways = ways;
    do {
        sw = ((tmpways << wshift) | (sets << sshift));
        SCB->DCCISW = sw;
    } while(tmpways--);
} while(sets--);

__DSB();
__ISB();
#endif
}

/***
\fn     void SCB_InvalidateDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief   D-Cache Invalidate by address
\param[in]  addr  address (aligned to 32-byte boundary)
\param[in]  dsize  size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_InvalidateDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;

```

```

    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;           // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

    __DSB();

    while (op_size > 0) {
        SCB->DCIMVAC = op_addr;
        op_addr +=     linesize;
        op_size -= (int32_t)linesize;
    }

    __DSB();
    __ISB();
#endif
}

/***
\fn      void SCB_CleanDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief   D-Cache Clean by address
\param[in] addr  address (aligned to 32-byte boundary)
\param[in] dsize size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_CleanDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;
    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;           // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

```

```

__DSB();

while (op_size > 0) {
    SCB->DCCMVAC = op_addr;
    op_addr +=     linesize;
    op_size -= (int32_t)linesize;
}

__DSB();
__ISB();
#endif
}

/***
\fn      void SCB_CleanInvalidateDCache_by_Addr(volatile uint32_t *addr, int32_t dsize)
\brief   D-Cache Clean and Invalidate by address
\param[in] addr  address (aligned to 32-byte boundary)
\param[in] dsize size of memory block (in number of bytes)
*/
__STATIC_INLINE void SCB_CleanInvalidateDCache_by_Addr (uint32_t *addr, int32_t dsize)
{
#if (__DCACHE_PRESENT == 1)
    int32_t op_size = dsize;
    uint32_t op_addr = (uint32_t)addr;
    uint32_t linesize = 32UL;          // in Cortex-M7 size of cache line is fixed to 8 words (32 bytes)

    __DSB();

```

```
while (op_size > 0) {  
    SCB->DCCIMVAC = op_addr;  
    op_addr +=     linesize;  
    op_size -= (int32_t)linesize;  
}  
  
__DSB();  
__ISB();  
#endif  
}  
  
/*@} end of CMSIS_Core_CacheFunctions */
```

```
/* ##### SysTick function  
##### */  
/** \ingroup CMSIS_Core_FunctionInterface  
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions  
 \brief Functions that configure the System.  
 @  
 */
```

```
#if (__Vendor_SysTickConfig == 0)  
  
/** \brief System Tick Configuration
```

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.

Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

```
\return 0 Function succeeded.
```

```
\return      1 Function failed.
```

\note When the variable **_Vendor_SysTickConfig** is set to 1, then the function **SysTick_Config** is not included. In this case, the file **device.h** must contain a vendor-specific implementation of this function.

*/

```

__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)

{
if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) { return (1UL); } /* Reload value impossible */

SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */

NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
*/

SysTick->VAL = 0UL; /* Load the SysTick Counter Value */

SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
    SysTick_CTRL_TICKINT_Msk |
    SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and SysTick Timer */

return (0UL); /* Function successful */

}

#endif

```

```

/*@} end of CMSIS_Core_SysTickFunctions */

/*
***** Debug In/Output function
***** */

/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_core_DebugFunctions ITM Functions
 \brief Functions that access the ITM debug interface.

 @{
 */

extern volatile int32_t ITM_RxBuffer;           /*!< External variable to receive characters.
 */

#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is
ready for next character. */

/** \brief ITM Send Character

The function transmits a character via the ITM channel 0, and
\li Just returns when no debugger is connected that has booked the output.
\li Is blocking when a debugger is connected, but the previous character sent has not been
transmitted.

\param [in] ch Character to transmit.

\returns Character to transmit.

*/
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)

```

```

{
if (((ITM->TCR & ITM_TCR_ITMENA_Msk) != 0UL) && /* ITM enabled */
    ((ITM->TER & 1UL ) != 0UL) ) /* ITM Port #0 enabled */

{
    while (ITM->PORT[0].u32 == 0UL) { __NOP(); }

    ITM->PORT[0].u8 = (uint8_t)ch;

}
return (ch);
}

```

/** \brief ITM Receive Character

The function inputs a character via the external variable \ref ITM_RxBuffer.

```

\return Received character.

\return -1 No character pending.

*/
__STATIC_INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1; /* no character available */

    if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
        ch = ITM_RxBuffer;
        ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
    }

    return (ch);
}

```

```
/** \brief ITM Check Character
```

The function checks whether a character is pending for reading in the variable \ref ITM_RxBuffer.

```
\return 0 No character available.  
\return 1 Character available.  
*/  
STATIC_INLINE int32_t ITM_CheckChar (void) {  
  
    if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {  
        return (0); /* no character available */  
    } else {  
        return (1); /* character available */  
    }  
}
```

```
/*@} end of CMSIS_core_DebugFunctions */
```

```
#ifdef __cplusplus  
}  
#endif
```

```
#endif /* __CORE_CM7_H_DEPENDANT */
```

```
#endif /* __CMSIS_GENERIC */
```

```
*****  
* * Authors: Harsimransingh Bindra, Smitesh Modak  
* * Date Edited: 10/29/2017  
* *  
* * File: uart.h  
* *  
* * Description: Header file for uart communication  
* * -UART_config  
* * -UART_send  
* * -UART_send_n  
* * -UART_recieve  
* * -UART_receive_n  
* * -UART0_IRQHandler  
* *  
* *  
*****  
header file for function called "uart"  
*****/
```

```
#ifndef INCLUDES_UART_H_  
#define INCLUDES_UART_H_
```

```
#include <stdio.h>  
#include <stdint.h>  
#include "MKL25Z4.h"  
#include "circbuf.h"
```

```

/*#if baud==115200
#define uart_reg->UART0_BDL 0x17
#define uart_reg->UART0_BDH 0x00
#define uart_reg->UART0_C4 0x0F

#elif baud==38400
#define uart_reg->UART0_BDL 0x44
#define uart_reg->UART0_BDH 0x00
#define uart_reg->UART0_C4 0x0F

#else baud==57200
#define uart_reg->UART0_BDL 0x2E
#define uart_reg->UART0_BDH 0x00
#define uart_reg->UART0_C4 0x0F

#endif*/

```

```

CB_t *create_buffer(CB_t *cbuf);
/***
* Create buffer with CB_t the struct defined in cirbuf.h
*
* This is used to create transmit and receive buffer.
*
*/

```

```

CB_status add_to_buffer(uint8_t *str_ptr);
/***
* Add items to the buffer.
*
*/

```

```
* This is used to add items to the buffer when transmitting and receiving. It uses CB_status enum to  
* give status about the buffer.  
*/
```

```
void UART_configure(void);  
/**  
 * Function to configure UART  
 *  
 * This function uses register masks to configure UART0 for receiving, transmitting and handling  
 interrupts.  
 * It enables MCGFLL clock, disables the uart initially, sets up the baud rate, enables Transmit and  
 Receive  
 * interrupts..  
 */
```

```
void UART_send(uint8_t *data);  
/**  
 * Function to transmit  
 *  
 * This function takes a pointer to the data. It puts the data in UART0_D register to transmit.  
 *  
 *  
 */
```

```
void UART_send_n(uint8_t *data, uint8_t length);  
/**  
 * Function to transmit data of length n  
 *  
 * This function takes pointer to the data and length of the data. It puts the data in UART0_D register  
 one by one
```

```
* to transmit.  
*  
*/  
  
void UART_receive(uint8_t *data);  
/**  
 * Function to receive data  
 *  
 * This function takes pointer to the data. It puts the data from UART0_D register to pointed address.  
 *  
 *  
 */  
  
void UART_receive_n(uint8_t *data,uint8_t length);  
/**  
 * Function to receive data of length n  
 *  
 * This function takes pointer to the data and the length of the data.  
 * It puts the data from UART0_D register to pointed address one by one.  
 *  
 *  
 */  
  
void UART0_IRQHandler();  
/**  
 * Function to handle interrupt.  
 *  
 * This function is invoked when the transmit and receive interrupt are enabled.
```

```
* It loads the data from the buffer into a variable and then transmits the data using send function,  
removes  
* the item. When the buffer count reaches 0, it disables the Transmit interrupt.  
* The received data is being stored in the buffer once the receive interrupt is on.  
*  
*/
```

```
#endif /* INCLUDES_UART_H_ */
```

```
/*  
** #####  
** Processors:      MKL25Z128FM4  
**                  MKL25Z128FT4  
**                  MKL25Z128LH4  
**                  MKL25Z128VLK4  
**  
** Compilers:       Keil ARM C/C++ Compiler  
**                  Freescale C/C++ for Embedded ARM  
**                  GNU C Compiler  
**                  GNU C Compiler - CodeSourcery Sourcery G++  
**                  IAR ANSI C/C++ Compiler for ARM  
**  
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012  
** Version:         rev. 2.5, 2015-02-19  
** Build:          b150220  
**  
** Abstract:
```

** CMSIS Peripheral Access Layer for MKL25Z4

**

** Copyright (c) 1997 - 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com

** mail: support@freescale.com

**

** Revisions:

** - rev. 1.0 (2012-06-13)

** Initial version.

** - rev. 1.1 (2012-06-21)

** Update according to reference manual rev. 1.

** - rev. 1.2 (2012-08-01)

** Device type UARTLP changed to UART0.

** - rev. 1.3 (2012-10-04)

** Update according to reference manual rev. 3.

** - rev. 1.4 (2012-11-22)

** MCG module - bit LOLS in MCG_S register renamed to LOLS0.

** NV registers - bit EZPORT_DIS in NV_FOPT register removed.

** - rev. 1.5 (2013-04-05)

** Changed start of doxygen comment.

** - rev. 2.0 (2013-10-29)

** Register accessor macros added to the memory map.

** Symbols for Processor Expert memory map compatibility added to the memory map.

** Startup file for gcc has been updated according to CMSIS 3.2.

** System initialization updated.

** - rev. 2.1 (2014-07-16)

** Module access macro module_BASES replaced by module_BASE_PTRS.

** System initialization and startup updated.

** - rev. 2.2 (2014-08-22)

** System initialization updated - default clock config changed.

** - rev. 2.3 (2014-08-28)

```
**      Update of startup files - possibility to override DefaultISR added.  
**      - rev. 2.4 (2014-10-14)  
**      Interrupt INT_LPTimer renamed to INT_LPTMR0.  
**      - rev. 2.5 (2015-02-19)  
**      Renamed interrupt vector LLW to LLWU.  
**  
** #####  
*/
```

```
/*!  
 * @file MKL25Z4.h  
 * @version 2.5  
 * @date 2015-02-19  
 * @brief CMSIS Peripheral Access Layer for MKL25Z4  
 *  
 * CMSIS Peripheral Access Layer for MKL25Z4  
*/
```

```
/* -----  
-- MCU activation  
----- */
```

```
/* Prevention from multiple including the same memory map */  
#if !defined(MKL25Z4_H_) /* Check if memory map has not been already included */  
#define MKL25Z4_H_  
#define MCU_MKL25Z4
```

```
/* Check if another memory map has not been also included */
```

```

#if (defined(MCU_ACTIVE))

#error MKL25Z4 memory map: There is already included another memory map. Only one memory map
can be included.

#endif /* (defined(MCU_ACTIVE)) */

#define MCU_ACTIVE

#include <stdint.h>

/** Memory map major version (memory maps with equal major version number are
 * compatible) */

#define MCU_MEM_MAP_VERSION 0x0200u

/** Memory map minor version */

#define MCU_MEM_MAP_VERSION_MINOR 0x0005u

/*
----- --
-- Interrupt vector numbers
----- */
/*!

* @addtogroup Interrupt_vector_numbers Interrupt vector numbers
* @{
*/
/*!

** Interrupt Number Definitions */

#define NUMBER_OF_INT_VECTORS 48           /**< Number of interrupts in the Vector table */

typedef enum IRQn {
    /* Core interrupts */

```

```

NonMaskableInt_IRQn      = -14,      /**< Non Maskable Interrupt */

HardFault_IRQn          = -13,      /**< Cortex-M0 SV Hard Fault Interrupt */

SVCall_IRQn             = -5,       /**< Cortex-M0 SV Call Interrupt */

PendSV_IRQn             = -2,       /**< Cortex-M0 Pend SV Interrupt */

SysTick_IRQn             = -1,       /**< Cortex-M0 System Tick Interrupt */

/* Device specific interrupts */

DMA0_IRQn               = 0,        /**< DMA channel 0 transfer complete */

DMA1_IRQn               = 1,        /**< DMA channel 1 transfer complete */

DMA2_IRQn               = 2,        /**< DMA channel 2 transfer complete */

DMA3_IRQn               = 3,        /**< DMA channel 3 transfer complete */

Reserved20_IRQn         = 4,        /**< Reserved interrupt */

FTFA_IRQn               = 5,        /**< Command complete and read collision */

LVD_LVW_IRQn            = 6,        /**< Low-voltage detect, low-voltage warning */

LLWU_IRQn               = 7,        /**< Low leakage wakeup Unit */

I2C0_IRQn               = 8,        /**< I2C0 interrupt */

I2C1_IRQn               = 9,        /**< I2C1 interrupt */

SPI0_IRQn               = 10,       /**< SPI0 single interrupt vector for all sources */

SPI1_IRQn               = 11,       /**< SPI1 single interrupt vector for all sources */

UART0_IRQn              = 12,       /**< UART0 status and error */

UART1_IRQn              = 13,       /**< UART1 status and error */

UART2_IRQn              = 14,       /**< UART2 status and error */

ADC0_IRQn               = 15,       /**< ADC0 interrupt */

CMP0_IRQn               = 16,       /**< CMP0 interrupt */

TPM0_IRQn               = 17,       /**< TPM0 single interrupt vector for all sources */

TPM1_IRQn               = 18,       /**< TPM1 single interrupt vector for all sources */

TPM2_IRQn               = 19,       /**< TPM2 single interrupt vector for all sources */

RTC_IRQn                = 20,       /**< RTC alarm */

RTC_Seconds_IRQn        = 21,       /**< RTC seconds */

```

```

PIT_IRQn          = 22,      /**< PIT interrupt */

Reserved39_IRQn = 23,      /**< Reserved interrupt */

USB0_IRQn        = 24,      /**< USB0 interrupt */

DAC0_IRQn        = 25,      /**< DAC0 interrupt */

TSI0_IRQn        = 26,      /**< TSI0 interrupt */

MCG_IRQn         = 27,      /**< MCG interrupt */

LPTMR0_IRQn     = 28,      /**< LPTMR0 interrupt */

Reserved45_IRQn = 29,      /**< Reserved interrupt */

PORTA_IRQn       = 30,      /**< PORTA Pin detect */

PORTD_IRQn       = 31,      /**< PORTD Pin detect */

} IRQn_Type;

```

```

/*!
* @}
*/
/* /* end of group Interrupt_vector_numbers */

```

```

/*
-- Cortex M0 Core Configuration
----- */

```

```

/*!
* @addtogroup Cortex_Core_Configuration Cortex M0 Core Configuration
* @{
*/

```

```

#define __CM0PLUS_REV      0x0000  /**< Core revision r0p0 */
#define __MPU_PRESENT       0      /**< Defines if an MPU is present or not */
#define __VTOR_PRESENT      1      /**< Defines if an MPU is present or not */

```

```
#define __NVIC_PRIO_BITS      2      /**< Number of priority bits implemented in the NVIC */  
#define __Vendor_SysTickConfig  0      /**< Vendor specific implementation of SysTickConfig is  
defined */  
  
#include "core_cm0plus.h"      /* Core Peripheral Access Layer */  
#include "system_MKL25Z4.h"    /* Device specific configuration file */  
  
/*!  
 * @}  
 */ /* end of group Cortex_Core_Configuration */
```

```
/* -----  
-- Device Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup Peripheral_access_layer Device Peripheral Access Layer  
 * @{  
 */
```

```
/*  
** Start of section using anonymous unions  
*/
```

```
#if defined(__ARMCC_VERSION)  
#pragma push  
#pragma anon_unions
```

```

#ifndef __ADC_H__
#define __ADC_H__

#include "fsl_registers.h"
#include "fsl_common.h"
#include "fsl_clock.h"
#include "fsl_iomux.h"
#include "fsl_adc.h"

/*-----*
-- ADC Peripheral Access Layer
-----*/
/*!
* @addtogroup ADC_Peripheral_Access_Layer ADC Peripheral Access Layer
* @{
*/
/* ** ADC - Register Layout Typedef */
typedef struct {
    __IO uint32_t SC1[2];           /* **< ADC Status and Control Registers 1, array offset: 0x0, array step: 0x4 */
    __IO uint32_t CFG1;             /* **< ADC Configuration Register 1, offset: 0x8 */
    __IO uint32_t CFG2;             /* **< ADC Configuration Register 2, offset: 0xC */
    __I  uint32_t R[2];              /* **< ADC Data Result Register, array offset: 0x10, array step: 0x4 */
    __IO uint32_t CV1;               /* **< Compare Value Registers, offset: 0x18 */
    __IO uint32_t CV2;               /* **< Compare Value Registers, offset: 0x1C */
} adc_regs_t;

```

```

__IO uint32_t SC2;           /**< Status and Control Register 2, offset: 0x20 */
__IO uint32_t SC3;           /**< Status and Control Register 3, offset: 0x24 */
__IO uint32_t OFS;           /**< ADC Offset Correction Register, offset: 0x28 */
__IO uint32_t PG;            /**< ADC Plus-Side Gain Register, offset: 0x2C */
__IO uint32_t MG;            /**< ADC Minus-Side Gain Register, offset: 0x30 */
__IO uint32_t CLPD;          /**< ADC Plus-Side General Calibration Value Register, offset:
0x34 */
__IO uint32_t CLPS;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x38
*/
__IO uint32_t CLP4;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x3C
*/
__IO uint32_t CLP3;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x40
*/
__IO uint32_t CLP2;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x44
*/
__IO uint32_t CLP1;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x48
*/
__IO uint32_t CLP0;          /**< ADC Plus-Side General Calibration Value Register, offset: 0x4C
*/
uint8_t RESERVED_0[4];
__IO uint32_t CLMD;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x54 */
__IO uint32_t CLMS;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x58 */
__IO uint32_t CLM4;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x5C */
__IO uint32_t CLM3;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x60 */
__IO uint32_t CLM2;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x64 */
__IO uint32_t CLM1;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x68 */
__IO uint32_t CLM0;          /**< ADC Minus-Side General Calibration Value Register, offset:
0x6C */

```

```

} ADC_Type, *ADC_MemMapPtr;

/*
-----  

-- ADC - Register accessor macros  

----- */
/*!  

* @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros  

* @{
* /  

  

/* ADC - Register accessors */  

#define ADC_SC1_REG(base,index) ((base)->SC1[index])  

#define ADC_SC1_COUNT 2  

#define ADC_CFG1_REG(base) ((base)->CFG1)  

#define ADC_CFG2_REG(base) ((base)->CFG2)  

#define ADC_R_REG(base,index) ((base)->R[index])  

#define ADC_R_COUNT 2  

#define ADC_CV1_REG(base) ((base)->CV1)  

#define ADC_CV2_REG(base) ((base)->CV2)  

#define ADC_SC2_REG(base) ((base)->SC2)  

#define ADC_SC3_REG(base) ((base)->SC3)  

#define ADC_OFS_REG(base) ((base)->OFS)  

#define ADC_PG_REG(base) ((base)->PG)  

#define ADC_MG_REG(base) ((base)->MG)  

#define ADC_CLPD_REG(base) ((base)->CLPD)  

#define ADC_CLPS_REG(base) ((base)->CLPS)  

#define ADC_CLP4_REG(base) ((base)->CLP4)

```

```
#define ADC_CLP3_REG(base) ((base)->CLP3)
#define ADC_CLP2_REG(base) ((base)->CLP2)
#define ADC_CLP1_REG(base) ((base)->CLP1)
#define ADC_CLP0_REG(base) ((base)->CLP0)
#define ADC_CLMD_REG(base) ((base)->CLMD)
#define ADC_CLMS_REG(base) ((base)->CLMS)
#define ADC_CLM4_REG(base) ((base)->CLM4)
#define ADC_CLM3_REG(base) ((base)->CLM3)
#define ADC_CLM2_REG(base) ((base)->CLM2)
#define ADC_CLM1_REG(base) ((base)->CLM1)
#define ADC_CLM0_REG(base) ((base)->CLM0)
```

```
/*
 * @{
 */
/* /* end of group ADC_Register_Accessor_Macros */
```

```
/*
-----  
-- ADC Register Masks  
----- */
```

```
/*
 * @addtogroup ADC_Register_Masks ADC Register Masks
 * @{
 */
/*
```

```
/* SC1 Bit Fields */

#define ADC_SC1_ADCH_MASK 0x1Fu
#define ADC_SC1_ADCH_SHIFT 0
```

```

#define ADC_SC1_ADCH_WIDTH          5
#define ADC_SC1_ADCH(x) (((uint32_t)((uint32_t)(x)<<ADC_SC1_ADCH_SHIFT))&ADC_SC1_ADCH_MASK)
#define ADC_SC1_DIFF_MASK           0x20u
#define ADC_SC1_DIFF_SHIFT          5
#define ADC_SC1_DIFF_WIDTH          1
#define ADC_SC1_DIFF(x) (((uint32_t)((uint32_t)(x)<<ADC_SC1_DIFF_SHIFT))&ADC_SC1_DIFF_MASK)
#define ADC_SC1_AIEN_MASK           0x40u
#define ADC_SC1_AIEN_SHIFT          6
#define ADC_SC1_AIEN_WIDTH          1
#define ADC_SC1_AIEN(x) (((uint32_t)((uint32_t)(x)<<ADC_SC1_AIEN_SHIFT))&ADC_SC1_AIEN_MASK)
#define ADC_SC1_COCO_MASK           0x80u
#define ADC_SC1_COCO_SHIFT          7
#define ADC_SC1_COCO_WIDTH          1
#define ADC_SC1_COCO(x) (((uint32_t)((uint32_t)(x)<<ADC_SC1_COCO_SHIFT))&ADC_SC1_COCO_MASK)

/* CFG1 Bit Fields */
#define ADC_CFG1_ADICLK_MASK         0x3u
#define ADC_CFG1_ADICLK_SHIFT        0
#define ADC_CFG1_ADICLK_WIDTH        2
#define ADC_CFG1_ADICLK(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADICLK_SHIFT))&ADC_CFG1_ADICLK_MASK)
#define ADC_CFG1_MODE_MASK           0xCu
#define ADC_CFG1_MODE_SHIFT          2
#define ADC_CFG1_MODE_WIDTH          2
#define ADC_CFG1_MODE(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG1_MODE_SHIFT))&ADC_CFG1_MODE_MASK)
#define ADC_CFG1_ADLSMP_MASK          0x10u
#define ADC_CFG1_ADLSMP_SHIFT         4

```

```

#define ADC_CFG1_ADLSMP_WIDTH          1
#define ADC_CFG1_ADLSMP(x)             (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADLSMP_SHIFT))&ADC_CFG1_ADLSMP_MASK)
#define ADC_CFG1_ADIV_MASK            0x60u
#define ADC_CFG1_ADIV_SHIFT           5
#define ADC_CFG1_ADIV_WIDTH           2
#define ADC_CFG1_ADIV(x)              (((uint32_t)((uint32_t)(x)<<ADC_CFG1_ADIV_SHIFT))&ADC_CFG1_ADIV_MASK)
#define ADC_CFG1_DLPC_MASK            0x80u
#define ADC_CFG1_DLPC_SHIFT           7
#define ADC_CFG1_DLPC_WIDTH           1
#define ADC_CFG1_DLPC(x)              (((uint32_t)((uint32_t)(x)<<ADC_CFG1_DLPC_SHIFT))&ADC_CFG1_DLPC_MASK)

/* CFG2 Bit Fields */

#define ADC_CFG2_DLSTS_MASK           0x3u
#define ADC_CFG2_DLSTS_SHIFT          0
#define ADC_CFG2_DLSTS_WIDTH          2
#define ADC_CFG2_DLSTS(x)             (((uint32_t)((uint32_t)(x)<<ADC_CFG2_DLSTS_SHIFT))&ADC_CFG2_DLSTS_MASK)
#define ADC_CFG2_ADHSC_MASK           0x4u
#define ADC_CFG2_ADHSC_SHIFT          2
#define ADC_CFG2_ADHSC_WIDTH          1
#define ADC_CFG2_ADHSC(x)             (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADHSC_SHIFT))&ADC_CFG2_ADHSC_MASK)
#define ADC_CFG2_ADACKEN_MASK         0x8u
#define ADC_CFG2_ADACKEN_SHIFT        3
#define ADC_CFG2_ADACKEN_WIDTH        1
#define ADC_CFG2_ADACKEN(x)           (((uint32_t)((uint32_t)(x)<<ADC_CFG2_ADACKEN_SHIFT))&ADC_CFG2_ADACKEN_MASK)
#define ADC_CFG2_MUXSEL_MASK          0x10u
#define ADC_CFG2_MUXSEL_SHIFT         4

```

```

#define ADC_CFG2_MUXSEL_WIDTH           1
#define ADC_CFG2_MUXSEL(x) (((uint32_t)((uint32_t)(x)<<ADC_CFG2_MUXSEL_SHIFT))&ADC_CFG2_MUXSEL_MASK)

/* R Bit Fields */

#define ADC_R_D_MASK                  0xFFFFu
#define ADC_R_D_SHIFT                 0
#define ADC_R_D_WIDTH                 16
#define ADC_R_D(x) ((((uint32_t)((uint32_t)(x)<<ADC_R_D_SHIFT))&ADC_R_D_MASK))

/* CV1 Bit Fields */

#define ADC_CV1_CV_MASK               0xFFFFu
#define ADC_CV1_CV_SHIFT              0
#define ADC_CV1_CV_WIDTH              16
#define ADC_CV1_CV(x) (((uint32_t)((uint32_t)(x)<<ADC_CV1_CV_SHIFT))&ADC_CV1_CV_MASK)

/* CV2 Bit Fields */

#define ADC_CV2_CV_MASK               0xFFFFu
#define ADC_CV2_CV_SHIFT              0
#define ADC_CV2_CV_WIDTH              16
#define ADC_CV2_CV(x) (((uint32_t)((uint32_t)(x)<<ADC_CV2_CV_SHIFT))&ADC_CV2_CV_MASK)

/* SC2 Bit Fields */

#define ADC_SC2_REFSEL_MASK            0x3u
#define ADC_SC2_REFSEL_SHIFT           0
#define ADC_SC2_REFSEL_WIDTH           2
#define ADC_SC2_REFSEL(x) (((uint32_t)((uint32_t)(x)<<ADC_SC2_REFSEL_SHIFT))&ADC_SC2_REFSEL_MASK)

#define ADC_SC2_DMAEN_MASK             0x4u
#define ADC_SC2_DMAEN_SHIFT            2
#define ADC_SC2_DMAEN_WIDTH            1

```

```

#define ADC_SC2_DMAEN(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_DMAEN_SHIFT))&ADC_SC2_DMAEN_MASK)

#define ADC_SC2_ACREN_MASK          0x8u
#define ADC_SC2_ACREN_SHIFT         3
#define ADC_SC2_ACREN_WIDTH         1

#define ADC_SC2_ACREN(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACREN_SHIFT))&ADC_SC2_ACREN_MASK)

#define ADC_SC2_ACFGTE_MASK         0x10u
#define ADC_SC2_ACFGTE_SHIFT        4
#define ADC_SC2_ACFGTE_WIDTH        1

#define ADC_SC2_ACFGTE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFGTE_SHIFT))&ADC_SC2_ACFGTE_MASK)

#define ADC_SC2_ACFE_MASK           0x20u
#define ADC_SC2_ACFE_SHIFT          5
#define ADC_SC2_ACFE_WIDTH          1

#define ADC_SC2_ACFE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ACFE_SHIFT))&ADC_SC2_ACFE_MASK)

#define ADC_SC2_ADTRG_MASK          0x40u
#define ADC_SC2_ADTRG_SHIFT         6
#define ADC_SC2_ADTRG_WIDTH         1

#define ADC_SC2_ADTRG(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADTRG_SHIFT))&ADC_SC2_ADTRG_MASK)

#define ADC_SC2_ADACT_MASK          0x80u
#define ADC_SC2_ADACT_SHIFT         7
#define ADC_SC2_ADACT_WIDTH         1

#define ADC_SC2_ADACT(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC2_ADACT_SHIFT))&ADC_SC2_ADACT_MASK)

/* SC3 Bit Fields */

#define ADC_SC3_AVGS_MASK           0x3u
#define ADC_SC3_AVGS_SHIFT          0
#define ADC_SC3_AVGS_WIDTH          2

```

```

#define ADC_SC3_AVGS(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGS_SHIFT))&ADC_SC3_AVGS_MASK)

#define ADC_SC3_AVGE_MASK          0x4u
#define ADC_SC3_AVGE_SHIFT        2
#define ADC_SC3_AVGE_WIDTH        1

#define ADC_SC3_AVGE(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_AVGE_SHIFT))&ADC_SC3_AVGE_MASK)

#define ADC_SC3_ADCO_MASK          0x8u
#define ADC_SC3_ADCO_SHIFT        3
#define ADC_SC3_ADCO_WIDTH        1

#define ADC_SC3_ADCO(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_ADCO_SHIFT))&ADC_SC3_ADCO_MASK)

#define ADC_SC3_CALF_MASK          0x40u
#define ADC_SC3_CALF_SHIFT        6
#define ADC_SC3_CALF_WIDTH        1

#define ADC_SC3_CALF(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CALF_SHIFT))&ADC_SC3_CALF_MASK)

#define ADC_SC3_CAL_MASK           0x80u
#define ADC_SC3_CAL_SHIFT         7
#define ADC_SC3_CAL_WIDTH         1

#define ADC_SC3_CAL(x)
(((uint32_t)((uint32_t)(x)<<ADC_SC3_CAL_SHIFT))&ADC_SC3_CAL_MASK)

/* OFS Bit Fields */

#define ADC_OFS_OFS_MASK           0xFFFFu
#define ADC_OFS_OFS_SHIFT          0
#define ADC_OFS_OFS_WIDTH         16

#define ADC_OFS_OFS(x)
(((uint32_t)((uint32_t)(x)<<ADC_OFS_OFS_SHIFT))&ADC_OFS_OFS_MASK)

/* PG Bit Fields */

#define ADC_PG_PG_MASK             0xFFFFu
#define ADC_PG_PG_SHIFT            0

```

```

#define ADC_PG_PG_WIDTH          16
#define ADC_PG_PG(x)             (((uint32_t)((uint32_t)(x))<<ADC_PG_PG_SHIFT))&ADC_PG_PG_MASK

/* MG Bit Fields */

#define ADC_MG_MG_MASK           0xFFFFu
#define ADC_MG_MG_SHIFT          0
#define ADC_MG_MG_WIDTH          16
#define ADC_MG_MG(x)              (((uint32_t)((uint32_t)(x))<<ADC_MG_MG_SHIFT))&ADC_MG_MG_MASK

/* CLPD Bit Fields */

#define ADC_CLPD_CLPD_MASK        0x3Fu
#define ADC_CLPD_CLPD_SHIFT       0
#define ADC_CLPD_CLPD_WIDTH       6
#define ADC_CLPD_CLPD(x)          (((uint32_t)((uint32_t)(x))<<ADC_CLPD_CLPD_SHIFT))&ADC_CLPD_CLPD_MASK

/* CLPS Bit Fields */

#define ADC_CLPS_CLPS_MASK        0x3Fu
#define ADC_CLPS_CLPS_SHIFT       0
#define ADC_CLPS_CLPS_WIDTH       6
#define ADC_CLPS_CLPS(x)          (((uint32_t)((uint32_t)(x))<<ADC_CLPS_CLPS_SHIFT))&ADC_CLPS_CLPS_MASK

/* CLP4 Bit Fields */

#define ADC_CLP4_CLP4_MASK         0x3FFu
#define ADC_CLP4_CLP4_SHIFT        0
#define ADC_CLP4_CLP4_WIDTH        10
#define ADC_CLP4_CLP4(x)           (((uint32_t)((uint32_t)(x))<<ADC_CLP4_CLP4_SHIFT))&ADC_CLP4_CLP4_MASK

/* CLP3 Bit Fields */

#define ADC_CLP3_CLP3_MASK         0x1FFu
#define ADC_CLP3_CLP3_SHIFT        0

```

```

#define ADC_CLP3_CLP3_WIDTH          9
#define ADC_CLP3_CLP3(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP3_CLP3_SHIFT))&ADC_CLP3_CLP3_MASK)

/* CLP2 Bit Fields */

#define ADC_CLP2_CLP2_MASK           0xFFu
#define ADC_CLP2_CLP2_SHIFT          0
#define ADC_CLP2_CLP2_WIDTH          8
#define ADC_CLP2_CLP2(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP2_CLP2_SHIFT))&ADC_CLP2_CLP2_MASK)

/* CLP1 Bit Fields */

#define ADC_CLP1_CLP1_MASK           0x7Fu
#define ADC_CLP1_CLP1_SHIFT          0
#define ADC_CLP1_CLP1_WIDTH          7
#define ADC_CLP1_CLP1(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP1_CLP1_SHIFT))&ADC_CLP1_CLP1_MASK)

/* CLP0 Bit Fields */

#define ADC_CLP0_CLP0_MASK           0x3Fu
#define ADC_CLP0_CLP0_SHIFT          0
#define ADC_CLP0_CLP0_WIDTH          6
#define ADC_CLP0_CLP0(x) (((uint32_t)((uint32_t)(x)<<ADC_CLP0_CLP0_SHIFT))&ADC_CLP0_CLP0_MASK)

/* CLMD Bit Fields */

#define ADC_CLMD_CLMD_MASK           0x3Fu
#define ADC_CLMD_CLMD_SHIFT          0
#define ADC_CLMD_CLMD_WIDTH          6
#define ADC_CLMD_CLMD(x) (((uint32_t)((uint32_t)(x)<<ADC_CLMD_CLMD_SHIFT))&ADC_CLMD_CLMD_MASK)

/* CLMS Bit Fields */

#define ADC_CLMS_CLMS_MASK           0x3Fu
#define ADC_CLMS_CLMS_SHIFT          0

```

```

#define ADC_CLMS_CLMS_WIDTH          6
#define ADC_CLMS_CLMS(x) (((uint32_t)((uint32_t)(x)<<ADC_CLMS_CLMS_SHIFT))&ADC_CLMS_CLMS_MASK)

/* CLM4 Bit Fields */

#define ADC_CLM4_CLM4_MASK           0x3FFu
#define ADC_CLM4_CLM4_SHIFT          0
#define ADC_CLM4_CLM4_WIDTH          10
#define ADC_CLM4_CLM4(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM4_CLM4_SHIFT))&ADC_CLM4_CLM4_MASK)

/* CLM3 Bit Fields */

#define ADC_CLM3_CLM3_MASK           0x1FFu
#define ADC_CLM3_CLM3_SHIFT          0
#define ADC_CLM3_CLM3_WIDTH          9
#define ADC_CLM3_CLM3(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM3_CLM3_SHIFT))&ADC_CLM3_CLM3_MASK)

/* CLM2 Bit Fields */

#define ADC_CLM2_CLM2_MASK           0xFFu
#define ADC_CLM2_CLM2_SHIFT          0
#define ADC_CLM2_CLM2_WIDTH          8
#define ADC_CLM2_CLM2(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM2_CLM2_SHIFT))&ADC_CLM2_CLM2_MASK)

/* CLM1 Bit Fields */

#define ADC_CLM1_CLM1_MASK           0x7Fu
#define ADC_CLM1_CLM1_SHIFT          0
#define ADC_CLM1_CLM1_WIDTH          7
#define ADC_CLM1_CLM1(x) (((uint32_t)((uint32_t)(x)<<ADC_CLM1_CLM1_SHIFT))&ADC_CLM1_CLM1_MASK)

/* CLM0 Bit Fields */

#define ADC_CLM0_CLM0_MASK           0x3Fu
#define ADC_CLM0_CLM0_SHIFT          0

```

```

#define ADC_CLM0_CLM0_WIDTH          6

#define ADC_CLM0_CLM0(x)
(((uint32_t)((uint32_t)(x)<<ADC_CLM0_CLM0_SHIFT))&ADC_CLM0_CLM0_MASK)

/*
 * @}
 */

/* /* end of group ADC_Register_Masks */

/* ADC - Peripheral instance base addresses */

/** Peripheral ADC0 base address */

#define ADC0_BASE                  (0x4003B000u)

/** Peripheral ADC0 base pointer */

#define ADC0                      ((ADC_Type *)ADC0_BASE)

#define ADC0_BASE_PTR              (ADC0)

/** Array initializer of ADC peripheral base addresses */

#define ADC_BASE_ADDRS            { ADC0_BASE }

/** Array initializer of ADC peripheral base pointers */

#define ADC_BASE_PTRS              { ADC0 }

/*
----- --
-- ADC - Register accessor macros
----- */

```

```

/*
 * @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor macros
 * @{
 */


```

```
/* ADC - Register instance definitions */

/* ADC0 */

#define ADC0_SC1A          ADC_SC1_REG(ADC0,0)
#define ADC0_SC1B          ADC_SC1_REG(ADC0,1)
#define ADC0_CFG1           ADC_CFG1_REG(ADC0)
#define ADC0_CFG2           ADC_CFG2_REG(ADC0)
#define ADC0_RA             ADC_R_REG(ADC0,0)
#define ADC0_RB             ADC_R_REG(ADC0,1)
#define ADC0_CV1             ADC_CV1_REG(ADC0)
#define ADC0_CV2             ADC_CV2_REG(ADC0)
#define ADC0_SC2             ADC_SC2_REG(ADC0)
#define ADC0_SC3             ADC_SC3_REG(ADC0)
#define ADC0_OFS             ADC_OFS_REG(ADC0)
#define ADC0_PG              ADC_PG_REG(ADC0)
#define ADC0_MG              ADC_MG_REG(ADC0)
#define ADC0_CLPD            ADC_CLPD_REG(ADC0)
#define ADC0_CLPS            ADC_CLPS_REG(ADC0)
#define ADC0_CLP4            ADC_CLP4_REG(ADC0)
#define ADC0_CLP3            ADC_CLP3_REG(ADC0)
#define ADC0_CLP2            ADC_CLP2_REG(ADC0)
#define ADC0_CLP1            ADC_CLP1_REG(ADC0)
#define ADC0_CLP0            ADC_CLP0_REG(ADC0)
#define ADC0_CLMD            ADC_CLMD_REG(ADC0)
#define ADC0_CLMS            ADC_CLMS_REG(ADC0)
#define ADC0_CLM4            ADC_CLM4_REG(ADC0)
#define ADC0_CLM3            ADC_CLM3_REG(ADC0)
#define ADC0_CLM2            ADC_CLM2_REG(ADC0)
#define ADC0_CLM1            ADC_CLM1_REG(ADC0)
```

```
#define ADC0_CLMO          ADC_CLMO_REG(ADC0)

/* ADC - Register array accessors */

#define ADC0_SC1(index)      ADC_SC1_REG(ADC0,index)
#define ADC0_R(index)        ADC_R_REG(ADC0,index)

/*!
 * @}

/* /* end of group ADC_Register_Accessor_Macros */
```

```
/*!
 * @}

/* /* end of group ADC_Peripheral_Access_Layer */
```

```
/* -----
-- CMP Peripheral Access Layer
----- */
```

```
/*!
 * @addtogroup CMP_Peripheral_Access_Layer CMP Peripheral Access Layer
 * @{
 * /
```

```
/** CMP - Register Layout Typedef */

typedef struct {

    __IO uint8_t CRO;           /**< CMP Control Register 0, offset: 0x0 */
    __IO uint8_t CR1;           /**< CMP Control Register 1, offset: 0x1 */
}
```

```
    __IO uint8_t FPR;           /**< CMP Filter Period Register, offset: 0x2 */
    __IO uint8_t SCR;          /**< CMP Status and Control Register, offset: 0x3 */
    __IO uint8_t DACCR;        /**< DAC Control Register, offset: 0x4 */
    __IO uint8_t MUXCR;        /**< MUX Control Register, offset: 0x5 */
} CMP_Type, *CMP_MemMapPtr;
```

```
/*
-----*
-- CMP - Register accessor macros
----- */
```

```
/*!
* @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros
* @{
*/

```

```
/* CMP - Register accessors */

#define CMP_CRO_REG(base)      ((base)->CR0)
#define CMP_CR1_REG(base)      ((base)->CR1)
#define CMP_FPR_REG(base)      ((base)->FPR)
#define CMP_SCR_REG(base)      ((base)->SCR)
#define CMP_DACCR_REG(base)    ((base)->DACCR)
#define CMP_MUXCR_REG(base)    ((base)->MUXCR)
```

```
/*
* @@
*/
/* end of group CMP_Register_Accessor_Macros */
```

```

/*
-----  

-- CMP Register Masks  

----- */
  

/*!  

 * @addtogroup CMP_Register_Masks CMP Register Masks
 * @{
 * /  

  

/* CRO Bit Fields */  

#define CMP_CRO_HYSTCTR_MASK          0x3u
#define CMP_CRO_HYSTCTR_SHIFT         0
#define CMP_CRO_HYSTCTR_WIDTH         2
#define CMP_CRO_HYSTCTR(x)            (((uint8_t)((uint8_t)(x))<<CMP_CRO_HYSTCTR_SHIFT))&CMP_CRO_HYSTCTR_MASK)  

#define CMP_CRO_FILTER_CNT_MASK        0x70u
#define CMP_CRO_FILTER_CNT_SHIFT       4
#define CMP_CRO_FILTER_CNT_WIDTH       3
#define CMP_CRO_FILTER_CNT(x)          (((uint8_t)((uint8_t)(x))<<CMP_CRO_FILTER_CNT_SHIFT))&CMP_CRO_FILTER_CNT_MASK)  

/* CR1 Bit Fields */  

#define CMP_CR1_EN_MASK               0x1u
#define CMP_CR1_EN_SHIFT              0
#define CMP_CR1_EN_WIDTH              1
#define CMP_CR1_EN(x)                 (((uint8_t)((uint8_t)(x))<<CMP_CR1_EN_SHIFT))&CMP_CR1_EN_MASK)  

#define CMP_CR1_OPE_MASK              0x2u
#define CMP_CR1_OPE_SHIFT             1
#define CMP_CR1_OPE_WIDTH             1

```

```

#define CMP_CR1_OPE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_OPE_SHIFT))&CMP_CR1_OPE_MASK)

#define CMP_CR1_COS_MASK          0x4u
#define CMP_CR1_COS_SHIFT         2
#define CMP_CR1_COS_WIDTH         1

#define CMP_CR1_COS(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_COS_SHIFT))&CMP_CR1_COS_MASK)

#define CMP_CR1_INV_MASK          0x8u
#define CMP_CR1_INV_SHIFT         3
#define CMP_CR1_INV_WIDTH         1

#define CMP_CR1_INV(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_INV_SHIFT))&CMP_CR1_INV_MASK)

#define CMP_CR1_PMODE_MASK         0x10u
#define CMP_CR1_PMODE_SHIFT        4
#define CMP_CR1_PMODE_WIDTH        1

#define CMP_CR1_PMODE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_PMODE_SHIFT))&CMP_CR1_PMODE_MASK)

#define CMP_CR1_TRIGM_MASK         0x20u
#define CMP_CR1_TRIGM_SHIFT        5
#define CMP_CR1_TRIGM_WIDTH        1

#define CMP_CR1_TRIGM(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_TRIGM_SHIFT))&CMP_CR1_TRIGM_MASK)

#define CMP_CR1_WE_MASK            0x40u
#define CMP_CR1_WE_SHIFT           6
#define CMP_CR1_WE_WIDTH           1

#define CMP_CR1_WE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_WE_SHIFT))&CMP_CR1_WE_MASK)

#define CMP_CR1_SE_MASK            0x80u
#define CMP_CR1_SE_SHIFT           7
#define CMP_CR1_SE_WIDTH           1

```

```

#define CMP_CR1_SE(x)
(((uint8_t)((uint8_t)(x)<<CMP_CR1_SE_SHIFT))&CMP_CR1_SE_MASK)

/* FPR Bit Fields */

#define CMP_FPR_filt_per_MASK          0xFFu
#define CMP_FPR_filt_per_SHIFT        0
#define CMP_FPR_filt_per_WIDTH        8

#define CMP_FPR_filt_per(x)
(((uint8_t)((uint8_t)(x)<<CMP_FPR_filt_per_SHIFT))&CMP_FPR_filt_per_MASK)

/* SCR Bit Fields */

#define CMP_SCR_cout_MASK            0x1u
#define CMP_SCR_cout_SHIFT          0
#define CMP_SCR_cout_WIDTH          1

#define CMP_SCR_cout(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_cout_SHIFT))&CMP_SCR_cout_MASK)

#define CMP_SCR_cff_MASK             0x2u
#define CMP_SCR_cff_SHIFT           1
#define CMP_SCR_cff_WIDTH           1

#define CMP_SCR_cff(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_cff_SHIFT))&CMP_SCR_cff_MASK)

#define CMP_SCR_cfr_MASK             0x4u
#define CMP_SCR_cfr_SHIFT           2
#define CMP_SCR_cfr_WIDTH           1

#define CMP_SCR_cfr(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_cfr_SHIFT))&CMP_SCR_cfr_MASK)

#define CMP_SCR_ief_MASK             0x8u
#define CMP_SCR_ief_SHIFT           3
#define CMP_SCR_ief_WIDTH           1

#define CMP_SCR_ief(x)
(((uint8_t)((uint8_t)(x)<<CMP_SCR_ief_SHIFT))&CMP_SCR_ief_MASK)

#define CMP_SCR_iер_MASK             0x10u
#define CMP_SCR_iер_SHIFT           4

```

```

#define CMP_SCR_IER_WIDTH          1
#define CMP_SCR_IER(x) (((uint8_t)((uint8_t)(x)<<CMP_SCR_IER_SHIFT))&CMP_SCR_IER_MASK)
#define CMP_SCR_DMAEN_MASK         0x40u
#define CMP_SCR_DMAEN_SHIFT        6
#define CMP_SCR_DMAEN_WIDTH        1
#define CMP_SCR_DMAEN(x) (((uint8_t)((uint8_t)(x)<<CMP_SCR_DMAEN_SHIFT))&CMP_SCR_DMAEN_MASK)

/* DACC Bit Fields */

#define CMP_DACCR_VOSEL_MASK       0x3Fu
#define CMP_DACCR_VOSEL_SHIFT      0
#define CMP_DACCR_VOSEL_WIDTH      6
#define CMP_DACCR_VOSEL(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_VOSEL_SHIFT))&CMP_DACCR_VOSEL_MASK)

#define CMP_DACCR_VRSEL_MASK        0x40u
#define CMP_DACCR_VRSEL_SHIFT       6
#define CMP_DACCR_VRSEL_WIDTH       1
#define CMP_DACCR_VRSEL(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_VRSEL_SHIFT))&CMP_DACCR_VRSEL_MASK)

#define CMP_DACCR_DACEN_MASK        0x80u
#define CMP_DACCR_DACEN_SHIFT       7
#define CMP_DACCR_DACEN_WIDTH       1
#define CMP_DACCR_DACEN(x) (((uint8_t)((uint8_t)(x)<<CMP_DACCR_DACEN_SHIFT))&CMP_DACCR_DACEN_MASK)

/* MUXCR Bit Fields */

#define CMP_MUXCR_MSEL_MASK         0x7u
#define CMP_MUXCR_MSEL_SHIFT        0
#define CMP_MUXCR_MSEL_WIDTH        3
#define CMP_MUXCR_MSEL(x) (((uint8_t)((uint8_t)(x)<<CMP_MUXCR_MSEL_SHIFT))&CMP_MUXCR_MSEL_MASK)

#define CMP_MUXCR_PSEL_MASK         0x38u

```

```

#define CMP_MUXCR_PSEL_SHIFT          3
#define CMP_MUXCR_PSEL_WIDTH          3
#define CMP_MUXCR_PSEL(x)
(((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSEL_SHIFT))&CMP_MUXCR_PSEL_MASK)

#define CMP_MUXCR_PSTM_MASK           0x80u
#define CMP_MUXCR_PSTM_SHIFT          7
#define CMP_MUXCR_PSTM_WIDTH          1
#define CMP_MUXCR_PSTM(x)
(((uint8_t)((uint8_t)(x)<<CMP_MUXCR_PSTM_SHIFT))&CMP_MUXCR_PSTM_MASK)

/*
 * @}
 */

/* /* end of group CMP_Register_Masks */

```

```

/* CMP - Peripheral instance base addresses */
/** Peripheral CMPO base address */
#define CMPO_BASE                   (0x40073000u)
/** Peripheral CMPO base pointer */
#define CMPO                         ((CMP_Type *)CMPO_BASE)
#define CMPO_BASE_PTR                (CMPO)
/** Array initializer of CMP peripheral base addresses */
#define CMP_BASE_ADDRS               { CMPO_BASE }
/** Array initializer of CMP peripheral base pointers */
#define CMP_BASE_PTRS                { CMPO }

```

```

/*
-----*
-- CMP - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor macros  
 * @{  
 */
```

```
/* CMP - Register instance definitions */  
/* CMPO */  
  
#define CMPO_CRO           CMP_CRO_REG(CMPO)  
#define CMPO_CR1           CMP_CR1_REG(CMPO)  
#define CMPO_FPR           CMP_FPR_REG(CMPO)  
#define CMPO_SCR           CMP_SCR_REG(CMPO)  
#define CMPO_DACCR         CMP_DACCR_REG(CMPO)  
#define CMPO_MUXCR         CMP_MUXCR_REG(CMPO)
```

```
/*!  
 * @}  
 */ /* end of group CMP_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group CMP_Peripheral_Access_Layer */
```

```
/* -----  
-- DAC Peripheral Access Layer  
----- */
```

```

/*!

* @addtogroup DAC_Peripheral_Access_Layer DAC Peripheral Access Layer

* @{

*/



/** DAC - Register Layout Typedef */

typedef struct {

    struct {                                     /* offset: 0x0, array step: 0x2 */
        __IO uint8_t DATL;                      /* < DAC Data Low Register, array offset: 0x0, array step: 0x2 */
        __IO uint8_t DATH;                      /* < DAC Data High Register, array offset: 0x1, array step: 0x2 */
    } DAT[2];

    uint8_t RESERVED_0[28];

    __IO uint8_t SR;                           /* < DAC Status Register, offset: 0x20 */
    __IO uint8_t C0;                           /* < DAC Control Register, offset: 0x21 */
    __IO uint8_t C1;                           /* < DAC Control Register 1, offset: 0x22 */
    __IO uint8_t C2;                           /* < DAC Control Register 2, offset: 0x23 */

} DAC_Type, *DAC_MemMapPtr;




/*
-----  

-- DAC - Register accessor macros  

----- */

```

```

/*!

* @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros

* @{

*/
```

```
/* DAC - Register accessors */

#define DAC_DATL_REG(base,index)      ((base)->DAT[index].DATL)
#define DAC_DATL_COUNT                2
#define DAC_DATH_REG(base,index)      ((base)->DAT[index].DATH)
#define DAC_DATH_COUNT                2
#define DAC_SR_REG(base)              ((base)->SR)
#define DAC_C0_REG(base)              ((base)->C0)
#define DAC_C1_REG(base)              ((base)->C1)
#define DAC_C2_REG(base)              ((base)->C2)

/*!
 * @}
 */
/* /* end of group DAC_Register_Accessor_Macros */
```

```
/*
-----  
-- DAC Register Masks  
----- */
```

```
/*!  
 * @addtogroup DAC_Register_Masks DAC Register Masks  
 * @{  
 */
```

```
/* DATL Bit Fields */

#define DAC_DATL_DATA0_MASK          0xFFu
#define DAC_DATL_DATA0_SHIFT         0
#define DAC_DATL_DATA0_WIDTH         8
```

```

#define DAC_DATL_DATA0(x)
(((uint8_t)((uint8_t)(x)<<DAC_DATL_DATA0_SHIFT))&DAC_DATL_DATA0_MASK)

/* DATH Bit Fields */

#define DAC_DATH_DATA1_MASK          0xFu
#define DAC_DATH_DATA1_SHIFT         0
#define DAC_DATH_DATA1_WIDTH         4

#define DAC_DATH_DATA1(x)
(((uint8_t)((uint8_t)(x)<<DAC_DATH_DATA1_SHIFT))&DAC_DATH_DATA1_MASK)

/* SR Bit Fields */

#define DAC_SR_DACBFRPBF_MASK        0x1u
#define DAC_SR_DACBFRPBF_SHIFT       0
#define DAC_SR_DACBFRPBF_WIDTH       1

#define DAC_SR_DACBFRPBF(x)
(((uint8_t)((uint8_t)(x)<<DAC_SR_DACBFRPBF_SHIFT))&DAC_SR_DACBFRPBF_MASK)

#define DAC_SR_DACBFRPTF_MASK        0x2u
#define DAC_SR_DACBFRPTF_SHIFT       1
#define DAC_SR_DACBFRPTF_WIDTH       1

#define DAC_SR_DACBFRPTF(x)
(((uint8_t)((uint8_t)(x)<<DAC_SR_DACBFRPTF_SHIFT))&DAC_SR_DACBFRPTF_MASK)

/* C0 Bit Fields */

#define DAC_C0_DACBBIEN_MASK         0x1u
#define DAC_C0_DACBBIEN_SHIFT        0
#define DAC_C0_DACBBIEN_WIDTH        1

#define DAC_C0_DACBBIEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C0_DACBBIEN_SHIFT))&DAC_C0_DACBBIEN_MASK)

#define DAC_C0_DACBTIEN_MASK         0x2u
#define DAC_C0_DACBTIEN_SHIFT        1
#define DAC_C0_DACBTIEN_WIDTH        1

#define DAC_C0_DACBTIEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C0_DACBTIEN_SHIFT))&DAC_C0_DACBTIEN_MASK)

#define DAC_C0_LPEN_MASK             0x8u

```

```

#define DAC_C0_LPEN_SHIFT          3
#define DAC_C0_LPEN_WIDTH          1
#define DAC_C0_LPEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_LPEN_SHIFT))&DAC_C0_LPEN_MASK)
#define DAC_C0_DACSWTRG_MASK        0x10u
#define DAC_C0_DACSWTRG_SHIFT       4
#define DAC_C0_DACSWTRG_WIDTH       1
#define DAC_C0_DACSWTRG(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACSWTRG_SHIFT))&DAC_C0_DACSWTRG_MASK)
#define DAC_C0_DACTRGSEL_MASK        0x20u
#define DAC_C0_DACTRGSEL_SHIFT       5
#define DAC_C0_DACTRGSEL_WIDTH       1
#define DAC_C0_DACTRGSEL(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACTRGSEL_SHIFT))&DAC_C0_DACTRGSEL_MASK)
#define DAC_C0_DACRFS_MASK           0x40u
#define DAC_C0_DACRFS_SHIFT          6
#define DAC_C0_DACRFS_WIDTH          1
#define DAC_C0_DACRFS(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACRFS_SHIFT))&DAC_C0_DACRFS_MASK)
#define DAC_C0_DACEN_MASK             0x80u
#define DAC_C0_DACEN_SHIFT            7
#define DAC_C0_DACEN_WIDTH            1
#define DAC_C0_DACEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C0_DACEN_SHIFT))&DAC_C0_DACEN_MASK)
/* C1 Bit Fields */
#define DAC_C1_DACBFEN_MASK           0x1u
#define DAC_C1_DACBFEN_SHIFT            0
#define DAC_C1_DACBFEN_WIDTH            1
#define DAC_C1_DACBFEN(x) (((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFEN_SHIFT))&DAC_C1_DACBFEN_MASK)
#define DAC_C1_DACBFMD_MASK           0x4u

```

```

#define DAC_C1_DACBFMD_SHIFT          2
#define DAC_C1_DACBFMD_WIDTH          1
#define DAC_C1_DACBFMD(x)
(((uint8_t)((uint8_t)(x)<<DAC_C1_DACBFMD_SHIFT))&DAC_C1_DACBFMD_MASK)

#define DAC_C1_DMAEN_MASK            0x80u
#define DAC_C1_DMAEN_SHIFT           7
#define DAC_C1_DMAEN_WIDTH           1
#define DAC_C1_DMAEN(x)
(((uint8_t)((uint8_t)(x)<<DAC_C1_DMAEN_SHIFT))&DAC_C1_DMAEN_MASK)

/* C2 Bit Fields */

#define DAC_C2_DACBFUP_MASK          0x1u
#define DAC_C2_DACBFUP_SHIFT          0
#define DAC_C2_DACBFUP_WIDTH          1
#define DAC_C2_DACBFUP(x)
(((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFUP_SHIFT))&DAC_C2_DACBFUP_MASK)

#define DAC_C2_DACBFRP_MASK          0x10u
#define DAC_C2_DACBFRP_SHIFT          4
#define DAC_C2_DACBFRP_WIDTH          1
#define DAC_C2_DACBFRP(x)
(((uint8_t)((uint8_t)(x)<<DAC_C2_DACBFRP_SHIFT))&DAC_C2_DACBFRP_MASK)

/*
 * @}
 */

/* /* end of group DAC_Register_Masks */

```

```

/* DAC - Peripheral instance base addresses */
/** Peripheral DAC0 base address */
#define DAC0_BASE                    (0x4003F000u)
/** Peripheral DAC0 base pointer */

```

```

#define DAC0 ((DAC_Type *)DAC0_BASE)

#define DAC0_BASE_PTR (DAC0)

/** Array initializer of DAC peripheral base addresses */

#define DAC_BASE_ADDRS { DAC0_BASE }

/** Array initializer of DAC peripheral base pointers */

#define DAC_BASE_PTRS { DAC0 }

/*
-----  

-- DAC - Register accessor macros  

----- */
/*!  

* @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor macros  

* @{
* /  

  

/* DAC - Register instance definitions */  

/* DAC0 */  

#define DAC0_DAT0L DAC_DATL_REG(DAC0,0)  

#define DAC0_DAT0H DAC_DATH_REG(DAC0,0)  

#define DAC0_DAT1L DAC_DATL_REG(DAC0,1)  

#define DAC0_DAT1H DAC_DATH_REG(DAC0,1)  

#define DAC0_SR DAC_SR_REG(DAC0)  

#define DAC0_C0 DAC_C0_REG(DAC0)  

#define DAC0_C1 DAC_C1_REG(DAC0)  

#define DAC0_C2 DAC_C2_REG(DAC0)  

  

/* DAC - Register array accessors */

```

```
#define DAC0_DATL(index)           DAC_DATL_REG(DAC0,index)
#define DAC0_DATH(index)           DAC_DATH_REG(DAC0,index)

/*!
 * @}
 */

/* /* end of group DAC_Register_Accessor_Macros */
```

```
/*!
 * @}
 */

/* /* end of group DAC_Peripheral_Access_Layer */
```

```
/*
-----  
-- DMA Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup DMA_Peripheral_Access_Layer DMA Peripheral Access Layer  
 * @{  
 */
```

```
/** DMA - Register Layout Typedef */  
  
typedef struct {  
    uint8_t RESERVED_0[256];  
  
    struct {  
        __IO uint32_t SAR;          /* offset: 0x100, array step: 0x10 */  
        /* < Source Address Register, array offset: 0x100, array step:  
        0x10 */  
    };
```

```

__IO uint32_t DAR;           /**< Destination Address Register, array offset: 0x104, array step:
0x10 */

union {                      /* offset: 0x108, array step: 0x10 */

    __IO uint32_t DSR_BCR;    /**< DMA Status Register / Byte Count Register, array offset:
0x108, array step: 0x10 */

    struct {                /* offset: 0x108, array step: 0x10 */

        uint8_t RESERVED_0[3];

        __IO uint8_t DSR;       /**< DMA_DSR0 register...DMA_DSR3 register., array offset:
0x10B, array step: 0x10 */

    } DMA_DSR_ACCESS8BIT;
};

__IO uint32_t DCR;           /**< DMA Control Register, array offset: 0x10C, array step: 0x10
*/
} DMA[4];

} DMA_Type, *DMA_MemMapPtr;

/*
-----  

-- DMA - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros  

* @{
* /

```

```

/* DMA - Register accessors */  

#define DMA_SAR_REG(base,index)      ((base)->DMA[index].SAR)  

#define DMA_SAR_COUNT               4  

#define DMA_DAR_REG(base,index)     ((base)->DMA[index].DAR)
```

```

#define DMA_DAR_COUNT           4
#define DMA_DSR_BCR_REG(base,index)      ((base)->DMA[index].DSR_BCR)
#define DMA_DSR_BCR_COUNT          4
#define DMA_DSR_REG(base,index)        ((base)->DMA[index].DMA_DSR_ACCESS8BIT.DSR)
#define DMA_DSR_COUNT              4
#define DMA_DCR_REG(base,index)        ((base)->DMA[index].DCR)
#define DMA_DCR_COUNT              4

/*!
 * @}
 */
/* /* end of group DMA_Register_Accessor_Macros */

```

```

/*
-----*
-- DMA Register Masks
----- */

```

```

/*!
 * @addtogroup DMA_Register_Masks DMA Register Masks
 * @{
 */

```

```

/* SAR Bit Fields */

#define DMA_SAR_SAR_MASK           0xFFFFFFFFu
#define DMA_SAR_SAR_SHIFT          0
#define DMA_SAR_SAR_WIDTH          32
#define DMA_SAR_SAR(x)             (((uint32_t)((uint32_t)(x)<<DMA_SAR_SAR_SHIFT))&DMA_SAR_SAR_MASK)

/* DAR Bit Fields */

```

```

#define DMA_DAR_DAR_MASK           0xFFFFFFFFu
#define DMA_DAR_DAR_SHIFT          0
#define DMA_DAR_DAR_WIDTH          32
#define DMA_DAR_DAR(x)             (((uint32_t)((uint32_t)(x))<<DMA_DAR_DAR_SHIFT))&DMA_DAR_DAR_MASK)

/* DSR_BCR Bit Fields */

#define DMA_DSR_BCR_BCR_MASK        0xFFFFFFFFu
#define DMA_DSR_BCR_BCR_SHIFT       0
#define DMA_DSR_BCR_BCR_WIDTH       24
#define DMA_DSR_BCR_BCR(x)          (((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_BCR_SHIFT))&DMA_DSR_BCR_BCR_MASK)

#define DMA_DSR_BCR_DONE_MASK        0x1000000u
#define DMA_DSR_BCR_DONE_SHIFT       24
#define DMA_DSR_BCR_DONE_WIDTH       1
#define DMA_DSR_BCR_DONE(x)          (((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_DONE_SHIFT))&DMA_DSR_BCR_DONE_MASK)

#define DMA_DSR_BCR_BSY_MASK        0x2000000u
#define DMA_DSR_BCR_BSY_SHIFT       25
#define DMA_DSR_BCR_BSY_WIDTH       1
#define DMA_DSR_BCR_BSY(x)          (((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_BSY_SHIFT))&DMA_DSR_BCR_BSY_MASK)

#define DMA_DSR_BCR_REQ_MASK        0x4000000u
#define DMA_DSR_BCR_REQ_SHIFT       26
#define DMA_DSR_BCR_REQ_WIDTH       1
#define DMA_DSR_BCR_REQ(x)          (((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_REQ_SHIFT))&DMA_DSR_BCR_REQ_MASK)

#define DMA_DSR_BCR_BED_MASK        0x10000000u
#define DMA_DSR_BCR_BED_SHIFT       28
#define DMA_DSR_BCR_BED_WIDTH       1
#define DMA_DSR_BCR_BED(x)          (((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_BED_SHIFT))&DMA_DSR_BCR_BED_MASK)

```

```

#define DMA_DSR_BCR_BES_MASK          0x20000000u
#define DMA_DSR_BCR_BES_SHIFT         29
#define DMA_DSR_BCR_BES_WIDTH         1
#define DMA_DSR_BCR_BES(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_BES_SHIFT))&DMA_DSR_BCR_BES_MASK)

#define DMA_DSR_BCR_CE_MASK          0x40000000u
#define DMA_DSR_BCR_CE_SHIFT         30
#define DMA_DSR_BCR_CE_WIDTH         1
#define DMA_DSR_BCR_CE(x)            (((uint32_t)((uint32_t)(x)<<DMA_DSR_BCR_CE_SHIFT))&DMA_DSR_BCR_CE_MASK)

/* DCR Bit Fields */

#define DMA_DCR_LCH2_MASK             0x3u
#define DMA_DCR_LCH2_SHIFT            0
#define DMA_DCR_LCH2_WIDTH            2
#define DMA_DCR_LCH2(x)               (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH2_SHIFT))&DMA_DCR_LCH2_MASK)

#define DMA_DCR_LCH1_MASK             0xCu
#define DMA_DCR_LCH1_SHIFT            2
#define DMA_DCR_LCH1_WIDTH            2
#define DMA_DCR_LCH1(x)               (((uint32_t)((uint32_t)(x)<<DMA_DCR_LCH1_SHIFT))&DMA_DCR_LCH1_MASK)

#define DMA_DCR_LINKCC_MASK           0x30u
#define DMA_DCR_LINKCC_SHIFT          4
#define DMA_DCR_LINKCC_WIDTH          2
#define DMA_DCR_LINKCC(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_LINKCC_SHIFT))&DMA_DCR_LINKCC_MASK)

#define DMA_DCR_D_REQ_MASK            0x80u
#define DMA_DCR_D_REQ_SHIFT           7
#define DMA_DCR_D_REQ_WIDTH           1
#define DMA_DCR_D_REQ(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_D_REQ_SHIFT))&DMA_DCR_D_REQ_MASK)

```

```

#define DMA_DCR_DMOD_MASK           0xF00u
#define DMA_DCR_DMOD_SHIFT          8
#define DMA_DCR_DMOD_WIDTH          4
#define DMA_DCR_DMOD(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_DMOD_SHIFT))&DMA_DCR_DMOD_MASK)

#define DMA_DCR_SMOD_MASK           0xF000u
#define DMA_DCR_SMOD_SHIFT          12
#define DMA_DCR_SMOD_WIDTH          4
#define DMA_DCR_SMOD(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_SMOD_SHIFT))&DMA_DCR_SMOD_MASK)

#define DMA_DCR_START_MASK          0x10000u
#define DMA_DCR_START_SHIFT          16
#define DMA_DCR_START_WIDTH          1
#define DMA_DCR_START(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_START_SHIFT))&DMA_DCR_START_MASK)

#define DMA_DCR_DSIZEMASK           0x60000u
#define DMA_DCR_DSIZE_SHIFT          17
#define DMA_DCR_DSIZEWIDTH           2
#define DMA_DCR_DSIZE(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_DSIZE_SHIFT))&DMA_DCR_DSIZEMASK)

#define DMA_DCR_DINC_MASK           0x80000u
#define DMA_DCR_DINC_SHIFT          19
#define DMA_DCR_DINC_WIDTH           1
#define DMA_DCR_DINC(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_DINC_SHIFT))&DMA_DCR_DINC_MASK)

#define DMA_DCR_SSIZEMASK           0x300000u
#define DMA_DCR_SSIZE_SHIFT          20
#define DMA_DCR_SSIZEWIDTH           2
#define DMA_DCR_SSIZE(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_SSIZE_SHIFT))&DMA_DCR_SSIZEMASK)

#define DMA_DCR_SINC_MASK           0x400000u

```

```

#define DMA_DCR_SINC_SHIFT          22
#define DMA_DCR_SINC_WIDTH          1
#define DMA_DCR_SINC(x)             (((uint32_t)((uint32_t)(x)<<DMA_DCR_SINC_SHIFT))&DMA_DCR_SINC_MASK)

#define DMA_DCR_EADREQ_MASK         0x800000u
#define DMA_DCR_EADREQ_SHIFT        23
#define DMA_DCR_EADREQ_WIDTH        1
#define DMA_DCR_EADREQ(x)           (((uint32_t)((uint32_t)(x)<<DMA_DCR_EADREQ_SHIFT))&DMA_DCR_EADREQ_MASK)

#define DMA_DCR_AA_MASK              0x10000000u
#define DMA_DCR_AA_SHIFT             28
#define DMA_DCR_AA_WIDTH             1
#define DMA_DCR_AA(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_AA_SHIFT))&DMA_DCR_AA_MASK)

#define DMA_DCR_CS_MASK              0x20000000u
#define DMA_DCR_CS_SHIFT             29
#define DMA_DCR_CS_WIDTH             1
#define DMA_DCR_CS(x)                (((uint32_t)((uint32_t)(x)<<DMA_DCR_CS_SHIFT))&DMA_DCR_CS_MASK)

#define DMA_DCR_ERQ_MASK             0x40000000u
#define DMA_DCR_ERQ_SHIFT            30
#define DMA_DCR_ERQ_WIDTH            1
#define DMA_DCR_ERQ(x)               (((uint32_t)((uint32_t)(x)<<DMA_DCR_ERQ_SHIFT))&DMA_DCR_ERQ_MASK)

#define DMA_DCR_EINT_MASK            0x80000000u
#define DMA_DCR_EINT_SHIFT           31
#define DMA_DCR_EINT_WIDTH           1
#define DMA_DCR_EINT(x)              (((uint32_t)((uint32_t)(x)<<DMA_DCR_EINT_SHIFT))&DMA_DCR_EINT_MASK)

```

/*!

```

* @}

/* /* end of group DMA_Register_Masks */

/* DMA - Peripheral instance base addresses */

/** Peripheral DMA base address */

#define DMA_BASE           (0x40008000u)

/** Peripheral DMA base pointer */

#define DMA0                ((DMA_Type *)DMA_BASE)

#define DMA_BASE_PTR        (DMA0)

/** Array initializer of DMA peripheral base addresses */

#define DMA_BASE_ADDRS     { DMA_BASE }

/** Array initializer of DMA peripheral base pointers */

#define DMA_BASE_PTRS       { DMA0 }

/*
-----  

-- DMA - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor macros  

* @{
* /  

  

/* DMA - Register instance definitions */  

/* DMA */  

#define DMA_SAR0            DMA_SAR_REG(DMA0,0)  

#define DMA_DAR0            DMA_DAR_REG(DMA0,0)

```

```

#define DMA_DSR_BCR0           DMA_DSR_BCR_REG(DMA0,0)
#define DMA_DSR0                DMA_DSR_REG(DMA0,0)
#define DMA_DCR0                DMA_DCR_REG(DMA0,0)
#define DMA_SAR1                DMA_SAR_REG(DMA0,1)
#define DMA_DAR1                DMA_DAR_REG(DMA0,1)
#define DMA_DSR_BCR1           DMA_DSR_BCR_REG(DMA0,1)
#define DMA_DSR1                DMA_DSR_REG(DMA0,1)
#define DMA_DCR1                DMA_DCR_REG(DMA0,1)
#define DMA_SAR2                DMA_SAR_REG(DMA0,2)
#define DMA_DAR2                DMA_DAR_REG(DMA0,2)
#define DMA_DSR_BCR2           DMA_DSR_BCR_REG(DMA0,2)
#define DMA_DSR2                DMA_DSR_REG(DMA0,2)
#define DMA_DCR2                DMA_DCR_REG(DMA0,2)
#define DMA_SAR3                DMA_SAR_REG(DMA0,3)
#define DMA_DAR3                DMA_DAR_REG(DMA0,3)
#define DMA_DSR_BCR3           DMA_DSR_BCR_REG(DMA0,3)
#define DMA_DSR3                DMA_DSR_REG(DMA0,3)
#define DMA_DCR3                DMA_DCR_REG(DMA0,3)

```

```

/* DMA - Register array accessors */

#define DMA_SAR(index)          DMA_SAR_REG(DMA0,index)
#define DMA_DAR(index)          DMA_DAR_REG(DMA0,index)
#define DMA_DSR_BCR(index)      DMA_DSR_BCR_REG(DMA0,index)
#define DMA_DSR(index)          DMA_DSR_REG(DMA0,index)
#define DMA_DCR(index)          DMA_DCR_REG(DMA0,index)

```

```

!*/
* @}

/* /* end of group DMA_Register_Accessor_Macros */

```

```

/*!
* @}
/* /* end of group DMA_Peripheral_Access_Layer */

/*
-----  

-- DMAMUX Peripheral Access Layer  

----- */
/*!  

* @addtogroup DMAMUX_Peripheral_Access_Layer DMAMUX Peripheral Access Layer
* @{
* /  

  

/** DMAMUX - Register Layout Typedef */
typedef struct {
    __IO uint8_t CHCFG[4];           /**< Channel Configuration register, array offset: 0x0, array step:  

0x1 */
} DMAMUX_Type, *DMAMUX_MemMapPtr;  

  

/*
-----  

-- DMAMUX - Register accessor macros  

----- */
/*!  

* @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros
* @{

```

```

*/



/* DMAMUX - Register accessors */

#define DMAMUX_CHCFG_REG(base,index)      ((base)->CHCFG[index])
#define DMAMUX_CHCFG_COUNT           4

/*!
 * @}

/* /* end of group DMAMUX_Register_Accessor_Macros */

/*
-----  

-- DMAMUX Register Masks  

----- */
/*!

* @addtogroup DMAMUX_Register_Masks DMAMUX Register Masks
* @{
* /


/* CHCFG Bit Fields */

#define DMAMUX_CHCFG_SOURCE_MASK      0x3Fu
#define DMAMUX_CHCFG_SOURCE_SHIFT    0
#define DMAMUX_CHCFG_SOURCE_WIDTH    6
#define DMAMUX_CHCFG_SOURCE(x)        (((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_SOURCE_SHIFT))&DMAMUX_CHCFG_SOURCE_MASK)

#define DMAMUX_CHCFG_TRIG_MASK       0x40u
#define DMAMUX_CHCFG_TRIG_SHIFT     6

```

```

#define DMAMUX_CHCFG_TRIG_WIDTH      1
#define DMAMUX_CHCFG_TRIG(x) (((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_TRIG_SHIFT))&DMAMUX_CHCFG_TRIG_MASK)
#define DMAMUX_CHCFG_ENBL_MASK        0x80u
#define DMAMUX_CHCFG_ENBL_SHIFT       7
#define DMAMUX_CHCFG_ENBL_WIDTH      1
#define DMAMUX_CHCFG_ENBL(x) (((uint8_t)((uint8_t)(x)<<DMAMUX_CHCFG_ENBL_SHIFT))&DMAMUX_CHCFG_ENBL_MASK)

/*
 * @}
 */
/* /* end of group DMAMUX_Register_Masks */

```

```

/* DMAMUX - Peripheral instance base addresses */
/** Peripheral DMAMUX0 base address */
#define DMAMUX0_BASE                (0x40021000u)
/** Peripheral DMAMUX0 base pointer */
#define DMAMUX0                      ((DMAMUX_Type *)DMAMUX0_BASE)
#define DMAMUX0_BASE_PTR             (DMAMUX0)
/** Array initializer of DMAMUX peripheral base addresses */
#define DMAMUX_BASE_ADDRS            { DMAMUX0_BASE }
/** Array initializer of DMAMUX peripheral base pointers */
#define DMAMUX_BASE_PTRS             { DMAMUX0 }

```

```

/*
-----  

-- DMAMUX - Register accessor macros  

----- */

```

```
/*!  
 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor macros  
 * @{  
 */
```

```
/* DMAMUX - Register instance definitions */  
  
/* DMAMUX0 */  
  
#define DMAMUX0_CHCFG0           DMAMUX_CHCFG_REG(DMAMUX0,0)  
#define DMAMUX0_CHCFG1           DMAMUX_CHCFG_REG(DMAMUX0,1)  
#define DMAMUX0_CHCFG2           DMAMUX_CHCFG_REG(DMAMUX0,2)  
#define DMAMUX0_CHCFG3           DMAMUX_CHCFG_REG(DMAMUX0,3)  
  
/* DMAMUX - Register array accessors */  
  
#define DMAMUX0_CHCFG(index)      DMAMUX_CHCFG_REG(DMAMUX0,index)
```

```
/*!  
 * @}  
 */ /* end of group DMAMUX_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group DMAMUX_Peripheral_Access_Layer */
```

```
/* -----  
-- FGPIO Peripheral Access Layer  
----- */
```

```

/*!

* @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access Layer

* @{

*/



/** GPIO - Register Layout Typedef */

typedef struct {

    __IO uint32_t PDOR;           /**< Port Data Output Register, offset: 0x0 */
    __O  uint32_t PSOR;          /**< Port Set Output Register, offset: 0x4 */
    __O  uint32_t PCOR;          /**< Port Clear Output Register, offset: 0x8 */
    __O  uint32_t PTOR;          /**< Port Toggle Output Register, offset: 0xC */
    __I  uint32_t PDIR;          /**< Port Data Input Register, offset: 0x10 */
    __IO uint32_t PDDR;          /**< Port Data Direction Register, offset: 0x14 */

} GPIO_Type, *GPIO_MemMapPtr;

/* -----
-- GPIO - Register accessor macros
----- */

/*!

* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros

* @{

*/



/* GPIO - Register accessors */

#define GPIO_PDOR_REG(base)      ((base)->PDOR)
#define GPIO_PSOR_REG(base)      ((base)->PSOR)

```

```
#define GPIO_PCOR_REG(base)      ((base)->PCOR)
#define GPIO_PTOR_REG(base)       ((base)->PTOR)
#define GPIO_PDIR_REG(base)       ((base)->PDIR)
#define GPIO_PDDR_REG(base)       ((base)->PDDR)
```

```
/*!
 * @}
 */
/* /* end of group GPIO_Register_Accessor_Macros */
```

```
/*
-----  
-- GPIO Register Masks  
----- */
```

```
/*!  
 * @addtogroup GPIO_Register_Masks GPIO Register Masks  
 * @{  
 */
```

```
/* PDOR Bit Fields */  
  
#define GPIO_PDOR PDO_MASK      0xFFFFFFFFu  
#define GPIO_PDOR PDO_SHIFT     0  
#define GPIO_PDOR PDO_WIDTH    32  
  
#define GPIO_PDOR PDO(x)  
((uint32_t)((uint32_t)(x)<<GPIO_PDOR PDO_SHIFT))&GPIO_PDOR PDO_MASK)
```

```
/* PSOR Bit Fields */  
  
#define GPIO_PSOR PTSO_MASK     0xFFFFFFFFu  
#define GPIO_PSOR PTSO_SHIFT    0  
#define GPIO_PSOR PTSO_WIDTH   32
```

```

#define GPIO_PSOR_PTSO(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PSOR_PTSO_SHIFT))&GPIO_PSOR_PTSO_MASK)

/* PCOR Bit Fields */

#define GPIO_PCOR_PTCO_MASK          0xFFFFFFFFu
#define GPIO_PCOR_PTCO_SHIFT         0
#define GPIO_PCOR_PTCO_WIDTH         32

#define GPIO_PCOR_PTCO(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)

/* PTOR Bit Fields */

#define GPIO_PTOR_PTTO_MASK          0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT         0
#define GPIO_PTOR_PTTO_WIDTH         32

#define GPIO_PTOR_PTTO(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/* PDIR Bit Fields */

#define GPIO_PDIR_PDI_MASK          0xFFFFFFFFu
#define GPIO_PDIR_PDI_SHIFT         0
#define GPIO_PDIR_PDI_WIDTH         32

#define GPIO_PDIR_PDI(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK          0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT         0
#define GPIO_PDDR_PDD_WIDTH         32

#define GPIO_PDDR_PDD(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

!*/
* @}
/* /* end of group GPIO_Register_Masks */

```

```

/* FGPIO - Peripheral instance base addresses */

/** Peripheral FGPIOA base address */

#define FGPIOA_BASE          (0xF80FF000u)

/** Peripheral FGPIOA base pointer */

#define FGPIOA             ((FGPIO_Type *)FGPIOA_BASE)

#define FGPIOA_BASE_PTR     (FGPIOA)

/** Peripheral FGPIOB base address */

#define FGPIOB_BASE          (0xF80FF040u)

/** Peripheral FGPIOB base pointer */

#define FGPIOB             ((FGPIO_Type *)FGPIOB_BASE)

#define FGPIOB_BASE_PTR     (FGPIOB)

/** Peripheral FGPIOC base address */

#define FGPIOC_BASE          (0xF80FF080u)

/** Peripheral FGPIOC base pointer */

#define FGPIOC             ((FGPIO_Type *)FGPIOC_BASE)

#define FGPIOC_BASE_PTR     (FGPIOC)

/** Peripheral FGPIOD base address */

#define FGPIOD_BASE          (0xF80FF0C0u)

/** Peripheral FGPIOD base pointer */

#define FGPIOD             ((FGPIO_Type *)FGPIOD_BASE)

#define FGPIOD_BASE_PTR     (FGPIOD)

/** Peripheral FGPIOE base address */

#define FGPIOE_BASE          (0xF80FF100u)

/** Peripheral FGPIOE base pointer */

#define FGPIOE             ((FGPIO_Type *)FGPIOE_BASE)

#define FGPIOE_BASE_PTR     (FGPIOE)

/** Array initializer of FGPIO peripheral base addresses */

```

```

#define GPIO_BASE_ADDRS          { GPIOA_BASE, GPIOB_BASE, GPIOC_BASE, GPIOD_BASE,
GPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */

#define GPIO_BASE_PTRS           { GPIOA, GPIOB, GPIOC, GPIOD, GPIOE }

/*
-----*
-- GPIO - Register accessor macros
----- */
/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{
*/
/* GPIO - Register instance definitions */

/* GPIOA */

#define GPIOA_PDOR                GPIO_PDOR_REG(GPIOA)
#define GPIOA_PSOR                GPIO_PSOR_REG(GPIOA)
#define GPIOA_PCOR                GPIO_PCOR_REG(GPIOA)
#define GPIOA_PTOR                GPIO_PTOR_REG(GPIOA)
#define GPIOA_PDIR                GPIO_PDIR_REG(GPIOA)
#define GPIOA_PDDR                GPIO_PDDR_REG(GPIOA)

/* GPIOB */

#define GPIOB_PDOR                GPIO_PDOR_REG(GPIOB)
#define GPIOB_PSOR                GPIO_PSOR_REG(GPIOB)
#define GPIOB_PCOR                GPIO_PCOR_REG(GPIOB)
#define GPIOB_PTOR                GPIO_PTOR_REG(GPIOB)
#define GPIOB_PDIR                GPIO_PDIR_REG(GPIOB)

```

```

#define FGPIOB_PDDR          GPIO_PDDR_REG(FGPIOB)
/* FGPIOC */

#define FGPIOC_PDOR          GPIO_PDOR_REG(FGPIOC)
#define FGPIOC_PSOR          GPIO_PSOR_REG(FGPIOC)
#define FGPIOC_PCOR          GPIO_PCOR_REG(FGPIOC)
#define FGPIOC_PTOR          GPIO_PTOR_REG(FGPIOC)
#define FGPIOC_PDIR          GPIO_PDIR_REG(FGPIOC)
#define FGPIOC_PDDR          GPIO_PDDR_REG(FGPIOC)

/* FGPIOD */

#define FGPIOD_PDOR          GPIO_PDOR_REG(FGPIOD)
#define FGPIOD_PSOR          GPIO_PSOR_REG(FGPIOD)
#define FGPIOD_PCOR          GPIO_PCOR_REG(FGPIOD)
#define FGPIOD_PTOR          GPIO_PTOR_REG(FGPIOD)
#define FGPIOD_PDIR          GPIO_PDIR_REG(FGPIOD)
#define FGPIOD_PDDR          GPIO_PDDR_REG(FGPIOD)

/* FGPIOE */

#define FGPIOE_PDOR          GPIO_PDOR_REG(FGIOE)
#define FGPIOE_PSOR          GPIO_PSOR_REG(FGIOE)
#define FGPIOE_PCOR          GPIO_PCOR_REG(FGIOE)
#define FGPIOE_PTOR          GPIO_PTOR_REG(FGIOE)
#define FGPIOE_PDIR          GPIO_PDIR_REG(FGIOE)
#define FGPIOE_PDDR          GPIO_PDDR_REG(FGIOE)

/*
 * @}
 */
/* /* end of group GPIO_Register_Accessor_Macros */

```

```
/*!
```

```

* @}

/* /* end of group FGPIO_Peripheral_Access_Layer */

/*
-----*
-- FTFA Peripheral Access Layer
----- */
/*!

* @addtogroup FTFA_Peripheral_Access_Layer FTFA Peripheral Access Layer

* @{

*/
/** FTFA - Register Layout Typedef */

typedef struct {

    __IO uint8_t FSTAT;           /**< Flash Status Register, offset: 0x0 */

    __IO uint8_t FCNFG;          /**< Flash Configuration Register, offset: 0x1 */

    __I  uint8_t FSEC;           /**< Flash Security Register, offset: 0x2 */

    __I  uint8_t FOPT;           /**< Flash Option Register, offset: 0x3 */

    __IO uint8_t FCCOB3;         /**< Flash Common Command Object Registers, offset: 0x4 */

    __IO uint8_t FCCOB2;         /**< Flash Common Command Object Registers, offset: 0x5 */

    __IO uint8_t FCCOB1;         /**< Flash Common Command Object Registers, offset: 0x6 */

    __IO uint8_t FCCOB0;         /**< Flash Common Command Object Registers, offset: 0x7 */

    __IO uint8_t FCCOB7;         /**< Flash Common Command Object Registers, offset: 0x8 */

    __IO uint8_t FCCOB6;         /**< Flash Common Command Object Registers, offset: 0x9 */

    __IO uint8_t FCCOB5;         /**< Flash Common Command Object Registers, offset: 0xA */

    __IO uint8_t FCCOB4;         /**< Flash Common Command Object Registers, offset: 0xB */

    __IO uint8_t FCCOBB;         /**< Flash Common Command Object Registers, offset: 0xC */

    __IO uint8_t FCCOBA;         /**< Flash Common Command Object Registers, offset: 0xD */
}

```

```

__IO uint8_t FCCOB9;           /**< Flash Common Command Object Registers, offset: 0xE */
__IO uint8_t FCCOB8;           /**< Flash Common Command Object Registers, offset: 0xF */
__IO uint8_t FPROT3;           /**< Program Flash Protection Registers, offset: 0x10 */
__IO uint8_t FPROT2;           /**< Program Flash Protection Registers, offset: 0x11 */
__IO uint8_t FPROT1;           /**< Program Flash Protection Registers, offset: 0x12 */
__IO uint8_t FPROTO;           /**< Program Flash Protection Registers, offset: 0x13 */

} FTFA_Type, *FTFA_MemMapPtr;

```

```

/*
-----*
-- FTFA - Register accessor macros
----- */

```

```

/*!
* @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros
* @{
*/

```

```

/* FTFA - Register accessors */

#define FTFA_FSTAT_REG(base)      ((base)->FSTAT)
#define FTFA_FCNG_REG(base)       ((base)->FCNFG)
#define FTFA_FSEC_REG(base)       ((base)->FSEC)
#define FTFA_FOPT_REG(base)       ((base)->FOPT)
#define FTFA_FCCOB3_REG(base)     ((base)->FCCOB3)
#define FTFA_FCCOB2_REG(base)     ((base)->FCCOB2)
#define FTFA_FCCOB1_REG(base)     ((base)->FCCOB1)
#define FTFA_FCCOB0_REG(base)     ((base)->FCCOB0)
#define FTFA_FCCOB7_REG(base)     ((base)->FCCOB7)
#define FTFA_FCCOB6_REG(base)     ((base)->FCCOB6)

```

```
#define FTFA_FCCOB5_REG(base)      ((base)->FCCOB5)
#define FTFA_FCCOB4_REG(base)      ((base)->FCCOB4)
#define FTFA_FCCOBB_REG(base)      ((base)->FCCOBB)
#define FTFA_FCCOBA_REG(base)      ((base)->FCCOBA)
#define FTFA_FCCOB9_REG(base)      ((base)->FCCOB9)
#define FTFA_FCCOB8_REG(base)      ((base)->FCCOB8)
#define FTFA_FPROT3_REG(base)      ((base)->FPROT3)
#define FTFA_FPROT2_REG(base)      ((base)->FPROT2)
#define FTFA_FPROT1_REG(base)      ((base)->FPROT1)
#define FTFA_FPROT0_REG(base)      ((base)->FPROT0)
```

```
/*
 * @}
 */
/* /* end of group FTFA_Register_Accessor_Macros */
```

```
/*
-----  
-- FTFA Register Masks  
----- */
```

```
/*
 * @addtogroup FTFA_Register_Masks FTFA Register Masks
 * @{
 */

```

```
/* FSTAT Bit Fields */

#define FTFA_FSTAT_MGSTAT0_MASK      0x1u
#define FTFA_FSTAT_MGSTAT0_SHIFT     0
#define FTFA_FSTAT_MGSTAT0_WIDTH     1
```

```

#define FTFA_FSTAT_MGSTAT0(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_MGSTAT0_SHIFT))&FTFA_FSTAT_MGSTAT0_MASK)

#define FTFA_FSTAT_FPVIOL_MASK          0x10u
#define FTFA_FSTAT_FPVIOL_SHIFT        4
#define FTFA_FSTAT_FPVIOL_WIDTH        1

#define FTFA_FSTAT_FPVIOL(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_FPVIOL_SHIFT))&FTFA_FSTAT_FPVIOL_MASK)

#define FTFA_FSTAT_ACCERR_MASK         0x20u
#define FTFA_FSTAT_ACCERR_SHIFT       5
#define FTFA_FSTAT_ACCERR_WIDTH       1

#define FTFA_FSTAT_ACCERR(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_ACCERR_SHIFT))&FTFA_FSTAT_ACCERR_MASK)

#define FTFA_FSTAT_RDCOLERR_MASK      0x40u
#define FTFA_FSTAT_RDCOLERR_SHIFT    6
#define FTFA_FSTAT_RDCOLERR_WIDTH    1

#define FTFA_FSTAT_RDCOLERR(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_RDCOLERR_SHIFT))&FTFA_FSTAT_RDCOLERR_MASK)

#define FTFA_FSTAT_CCIF_MASK          0x80u
#define FTFA_FSTAT_CCIF_SHIFT        7
#define FTFA_FSTAT_CCIF_WIDTH        1

#define FTFA_FSTAT_CCIF(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSTAT_CCIF_SHIFT))&FTFA_FSTAT_CCIF_MASK)

/* FCNFG Bit Fields */

#define FTFA_FCNFG_ERSSUSP_MASK      0x10u
#define FTFA_FCNFG_ERSSUSP_SHIFT    4
#define FTFA_FCNFG_ERSSUSP_WIDTH    1

#define FTFA_FCNFG_ERSSUSP(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCNFG_ERSSUSP_SHIFT))&FTFA_FCNFG_ERSSUSP_MASK)

#define FTFA_FCNFG_ERSAREQ_MASK      0x20u
#define FTFA_FCNFG_ERSAREQ_SHIFT    5
#define FTFA_FCNFG_ERSAREQ_WIDTH    1

```

```

#define FTFA_FCNFG_ERSAREQ(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_ERSAREQ_SHIFT))&FTFA_FCNFG_ERSAREQ_MASK)

#define FTFA_FCNFG_RDCOLLIE_MASK          0x40u
#define FTFA_FCNFG_RDCOLLIE_SHIFT        6
#define FTFA_FCNFG_RDCOLLIE_WIDTH       1

#define FTFA_FCNFG_RDCOLLIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_RDCOLLIE_SHIFT))&FTFA_FCNFG_RDCOLLIE_MASK)

#define FTFA_FCNFG_CCIE_MASK            0x80u
#define FTFA_FCNFG_CCIE_SHIFT          7
#define FTFA_FCNFG_CCIE_WIDTH         1

#define FTFA_FCNFG_CCIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_CCIE_SHIFT))&FTFA_FCNFG_CCIE_MASK)

/* FSEC Bit Fields */

#define FTFA_FSEC_SEC_MASK             0x3u
#define FTFA_FSEC_SEC_SHIFT            0
#define FTFA_FSEC_SEC_WIDTH           2

#define FTFA_FSEC_SEC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_SEC_SHIFT))&FTFA_FSEC_SEC_MASK)

#define FTFA_FSEC_FSLACC_MASK          0xCu
#define FTFA_FSEC_FSLACC_SHIFT        2
#define FTFA_FSEC_FSLACC_WIDTH       2

#define FTFA_FSEC_FSLACC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_FSLACC_SHIFT))&FTFA_FSEC_FSLACC_MASK)

#define FTFA_FSEC_MEEN_MASK            0x30u
#define FTFA_FSEC_MEEN_SHIFT          4
#define FTFA_FSEC_MEEN_WIDTH         2

#define FTFA_FSEC_MEEN(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_MEEN_SHIFT))&FTFA_FSEC_MEEN_MASK)

#define FTFA_FSEC_KEYEN_MASK           0xC0u
#define FTFA_FSEC_KEYEN_SHIFT          6
#define FTFA_FSEC_KEYEN_WIDTH         2

```

```

#define FTFA_FSEC_KEYEN(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FSEC_KEYEN_SHIFT))&FTFA_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define FTFA_FOPT_OPT_MASK          0xFFu
#define FTFA_FOPT_OPT_SHIFT         0
#define FTFA_FOPT_OPT_WIDTH         8

#define FTFA_FOPT_OPT(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FOPT_OPT_SHIFT))&FTFA_FOPT_OPT_MASK)

/* FCCOB3 Bit Fields */

#define FTFA_FCCOB3_CCOBn_MASK      0xFFu
#define FTFA_FCCOB3_CCOBn_SHIFT     0
#define FTFA_FCCOB3_CCOBn_WIDTH     8

#define FTFA_FCCOB3_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB3_CCOBn_SHIFT))&FTFA_FCCOB3_CCOBn_MASK)

/* FCCOB2 Bit Fields */

#define FTFA_FCCOB2_CCOBn_MASK      0xFFu
#define FTFA_FCCOB2_CCOBn_SHIFT     0
#define FTFA_FCCOB2_CCOBn_WIDTH     8

#define FTFA_FCCOB2_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB2_CCOBn_SHIFT))&FTFA_FCCOB2_CCOBn_MASK)

/* FCCOB1 Bit Fields */

#define FTFA_FCCOB1_CCOBn_MASK      0xFFu
#define FTFA_FCCOB1_CCOBn_SHIFT     0
#define FTFA_FCCOB1_CCOBn_WIDTH     8

#define FTFA_FCCOB1_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB1_CCOBn_SHIFT))&FTFA_FCCOB1_CCOBn_MASK)

/* FCCOB0 Bit Fields */

#define FTFA_FCCOB0_CCOBn_MASK      0xFFu
#define FTFA_FCCOB0_CCOBn_SHIFT     0
#define FTFA_FCCOB0_CCOBn_WIDTH     8

```

```

#define FTFA_FCCOB0_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB0_CCOBn_SHIFT))&FTFA_FCCOB0_CCOBn_MASK)

/* FCCOB7 Bit Fields */

#define FTFA_FCCOB7_CCOBn_MASK          0xFFu
#define FTFA_FCCOB7_CCOBn_SHIFT         0
#define FTFA_FCCOB7_CCOBn_WIDTH         8

#define FTFA_FCCOB7_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB7_CCOBn_SHIFT))&FTFA_FCCOB7_CCOBn_MASK)

/* FCCOB6 Bit Fields */

#define FTFA_FCCOB6_CCOBn_MASK          0xFFu
#define FTFA_FCCOB6_CCOBn_SHIFT         0
#define FTFA_FCCOB6_CCOBn_WIDTH         8

#define FTFA_FCCOB6_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB6_CCOBn_SHIFT))&FTFA_FCCOB6_CCOBn_MASK)

/* FCCOB5 Bit Fields */

#define FTFA_FCCOB5_CCOBn_MASK          0xFFu
#define FTFA_FCCOB5_CCOBn_SHIFT         0
#define FTFA_FCCOB5_CCOBn_WIDTH         8

#define FTFA_FCCOB5_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB5_CCOBn_SHIFT))&FTFA_FCCOB5_CCOBn_MASK)

/* FCCOB4 Bit Fields */

#define FTFA_FCCOB4_CCOBn_MASK          0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT         0
#define FTFA_FCCOB4_CCOBn_WIDTH         8

#define FTFA_FCCOB4_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK)

/* FCCOBB Bit Fields */

#define FTFA_FCCOBB_CCOBn_MASK          0xFFu
#define FTFA_FCCOBB_CCOBn_SHIFT         0
#define FTFA_FCCOBB_CCOBn_WIDTH         8

```

```

#define FTFA_FCCOBB_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOBB_CCOBn_SHIFT))&FTFA_FCCOBB_CCOBn_MASK)

/* FCCOBA Bit Fields */

#define FTFA_FCCOBA_CCOBn_MASK          0xFFu
#define FTFA_FCCOBA_CCOBn_SHIFT         0
#define FTFA_FCCOBA_CCOBn_WIDTH         8

#define FTFA_FCCOBA_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOBA_CCOBn_SHIFT))&FTFA_FCCOBA_CCOBn_MASK)

/* FCCOB9 Bit Fields */

#define FTFA_FCCOB9_CCOBn_MASK          0xFFu
#define FTFA_FCCOB9_CCOBn_SHIFT         0
#define FTFA_FCCOB9_CCOBn_WIDTH         8

#define FTFA_FCCOB9_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB9_CCOBn_SHIFT))&FTFA_FCCOB9_CCOBn_MASK)

/* FCCOB8 Bit Fields */

#define FTFA_FCCOB8_CCOBn_MASK          0xFFu
#define FTFA_FCCOB8_CCOBn_SHIFT         0
#define FTFA_FCCOB8_CCOBn_WIDTH         8

#define FTFA_FCCOB8_CCOBn(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FCCOB8_CCOBn_SHIFT))&FTFA_FCCOB8_CCOBn_MASK)

/* FPROT3 Bit Fields */

#define FTFA_FPROT3_PROT_MASK           0xFFu
#define FTFA_FPROT3_PROT_SHIFT          0
#define FTFA_FPROT3_PROT_WIDTH          8

#define FTFA_FPROT3_PROT(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

#define FTFA_FPROT2_PROT_MASK           0xFFu
#define FTFA_FPROT2_PROT_SHIFT          0
#define FTFA_FPROT2_PROT_WIDTH          8

```

```

#define FTFA_FPROT2_PROT(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FPROT2_PROT_SHIFT))&FTFA_FPROT2_PROT_MASK)

/* FPROT1 Bit Fields */

#define FTFA_FPROT1_PROT_MASK          0xFFu
#define FTFA_FPROT1_PROT_SHIFT         0
#define FTFA_FPROT1_PROT_WIDTH         8

#define FTFA_FPROT1_PROT(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FPROT1_PROT_SHIFT))&FTFA_FPROT1_PROT_MASK)

/* FPROT0 Bit Fields */

#define FTFA_FPROT0_PROT_MASK          0xFFu
#define FTFA_FPROT0_PROT_SHIFT         0
#define FTFA_FPROT0_PROT_WIDTH         8

#define FTFA_FPROT0_PROT(x)
(((uint8_t)((uint8_t)(x)<<FTFA_FPROT0_PROT_SHIFT))&FTFA_FPROT0_PROT_MASK)

/*
 * @}
 */

/* /* end of group FTFA_Register_Masks */

```

```

/* FTFA - Peripheral instance base addresses */

/** Peripheral FTFA base address */

#define FTFA_BASE          (0x40020000u)

/** Peripheral FTFA base pointer */

#define FTFA          ((FTFA_Type *)FTFA_BASE)

#define FTFA_BASE_PTR      (FTFA)

/** Array initializer of FTFA peripheral base addresses */

#define FTFA_BASE_ADDRS     { FTFA_BASE }

/** Array initializer of FTFA peripheral base pointers */

#define FTFA_BASE_PTRS      { FTFA }

```

```
/*
-----  
-- FTFA - Register accessor macros  
----- */  
  
/*!  
 * @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor macros  
 * @{  
 */  
  
/* FTFA - Register instance definitions */  
/* FTFA */  
  
#define FTFA_FSTAT           FTFA_FSTAT_REG(FTFA)  
#define FTFA_FCNGF           FTFA_FCNGF_REG(FTFA)  
#define FTFA_FSEC             FTFA_FSEC_REG(FTFA)  
#define FTFA_FOPT             FTFA_FOPT_REG(FTFA)  
#define FTFA_FCCOB3           FTFA_FCCOB3_REG(FTFA)  
#define FTFA_FCCOB2           FTFA_FCCOB2_REG(FTFA)  
#define FTFA_FCCOB1           FTFA_FCCOB1_REG(FTFA)  
#define FTFA_FCCOB0           FTFA_FCCOB0_REG(FTFA)  
#define FTFA_FCCOB7           FTFA_FCCOB7_REG(FTFA)  
#define FTFA_FCCOB6           FTFA_FCCOB6_REG(FTFA)  
#define FTFA_FCCOB5           FTFA_FCCOB5_REG(FTFA)  
#define FTFA_FCCOB4           FTFA_FCCOB4_REG(FTFA)  
#define FTFA_FCCOBB           FTFA_FCCOBB_REG(FTFA)  
#define FTFA_FCCOBA           FTFA_FCCOBA_REG(FTFA)  
#define FTFA_FCCOB9           FTFA_FCCOB9_REG(FTFA)  
#define FTFA_FCCOB8           FTFA_FCCOB8_REG(FTFA)
```

```

#define FTFA_FPROT3           FTFA_FPROT3_REG(FTFA)
#define FTFA_FPROT2           FTFA_FPROT2_REG(FTFA)
#define FTFA_FPROT1           FTFA_FPROT1_REG(FTFA)
#define FTFA_FPROT0           FTFA_FPROT0_REG(FTFA)

/*!
 * @}
 */

/* /* end of group FTFA_Register_Accessor_Macros */

/*!
 * @}
 */

/* /* end of group FTFA_Peripheral_Access_Layer */

/*
-----  

-- GPIO Peripheral Access Layer  

----- */
/*!  

 * @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access Layer
 * @{
 */  

  

/** GPIO - Register Layout Typedef */  

typedef struct {  

    __IO uint32_t PDOR;          /**< Port Data Output Register, offset: 0x0 */  

    __O uint32_t PSOR;          /**< Port Set Output Register, offset: 0x4 */  

    __O uint32_t PCOR;          /**< Port Clear Output Register, offset: 0x8 */
}

```

```
    __O uint32_t PTOR;           /**< Port Toggle Output Register, offset: 0xC */
    __I uint32_t PDIR;           /**< Port Data Input Register, offset: 0x10 */
    __IO uint32_t PDDR;          /**< Port Data Direction Register, offset: 0x14 */
}
} GPIO_Type, *GPIO_MemMapPtr;
```

```
/*
-----*
-- GPIO - Register accessor macros
----- */
```

```
/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{
*/

```

```
/* GPIO - Register accessors */
#define GPIO_PDOR_REG(base)      ((base)->PDOR)
#define GPIO_PSOR_REG(base)      ((base)->PSOR)
#define GPIO_PCOR_REG(base)      ((base)->PCOR)
#define GPIO_PTOR_REG(base)      ((base)->PTOR)
#define GPIO_PDIR_REG(base)      ((base)->PDIR)
#define GPIO_PDDR_REG(base)      ((base)->PDDR)
```

```
/*!
* @}
*/
/* /* end of group GPIO_Register_Accessor_Macros */
```

```
/*
-----*
```

```

-- GPIO Register Masks
----- */

/*!
* @addtogroup GPIO_Register_Masks GPIO Register Masks
* @{
*/
/* PDOR Bit Fields */
#define GPIO_PDOR_PDO_MASK          0xFFFFFFFFu
#define GPIO_PDOR_PDO_SHIFT         0
#define GPIO_PDOR_PDO_WIDTH         32
#define GPIO_PDOR_PDO(x)            (((uint32_t)((uint32_t)(x)<<GPIO_PDOR_PDO_SHIFT))&GPIO_PDOR_PDO_MASK)

/* PSOR Bit Fields */
#define GPIO_PSOR PTSO_MASK          0xFFFFFFFFu
#define GPIO_PSOR PTSO_SHIFT         0
#define GPIO_PSOR PTSO_WIDTH         32
#define GPIO_PSOR PTSO(x)            (((uint32_t)((uint32_t)(x)<<GPIO_PSOR_PTSO_SHIFT))&GPIO_PSOR_PTSO_MASK)

/* PCOR Bit Fields */
#define GPIO_PCOR_PTCO_MASK          0xFFFFFFFFu
#define GPIO_PCOR_PTCO_SHIFT         0
#define GPIO_PCOR_PTCO_WIDTH         32
#define GPIO_PCOR_PTCO(x)            (((uint32_t)((uint32_t)(x)<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)

/* PTOR Bit Fields */
#define GPIO_PTOR_PTTO_MASK          0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT         0
#define GPIO_PTOR_PTTO_WIDTH         32

```

```

#define GPIO_PTOR_PTTO(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)

/* PDIR Bit Fields */

#define GPIO_PDIR_PDI_MASK          0xFFFFFFFFu
#define GPIO_PDIR_PDI_SHIFT         0
#define GPIO_PDIR_PDI_WIDTH         32

#define GPIO_PDIR_PDI(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)

/* PDDR Bit Fields */

#define GPIO_PDDR_PDD_MASK          0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT         0
#define GPIO_PDDR_PDD_WIDTH         32

#define GPIO_PDDR_PDD(x)
(((uint32_t)((uint32_t)(x)<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

/*
 * @}
 */

/* /* end of group GPIO_Register_Masks */

```

```

/* GPIO - Peripheral instance base addresses */

/** Peripheral GPIOA base address */

#define GPIOA_BASE                  (0x400FF000u)

/** Peripheral GPIOA base pointer */

#define GPIOA                      ((GPIO_Type *)GPIOA_BASE)

#define GPIOA_BASE_PTR              (GPIOA)

/** Peripheral GPIOB base address */

#define GPIOB_BASE                  (0x400FF040u)

/** Peripheral GPIOB base pointer */

#define GPIOB                      ((GPIO_Type *)GPIOB_BASE)

```

```

#define GPIOB_BASE_PTR          (GPIOB)
/** Peripheral GPIOC base address */

#define GPIOC_BASE              (0x400FF080u)
/** Peripheral GPIOC base pointer */

#define GPIOC                   ((GPIO_Type *)GPIOC_BASE)

#define GPIOC_BASE_PTR          (GPIOC)
/** Peripheral GPIOD base address */

#define GPIOD_BASE              (0x400FF0C0u)
/** Peripheral GPIOD base pointer */

#define GPIOD                   ((GPIO_Type *)GPIOD_BASE)

#define GPIOD_BASE_PTR          (GPIOD)
/** Peripheral GPIOE base address */

#define GPIOE_BASE              (0x400FF100u)
/** Peripheral GPIOE base pointer */

#define GPIOE                   ((GPIO_Type *)GPIOE_BASE)

#define GPIOE_BASE_PTR          (GPIOE)
/** Array initializer of GPIO peripheral base addresses */

#define GPIO_BASE_ADDRS         { GPIOA_BASE, GPIOB_BASE, GPIOC_BASE, GPIOD_BASE,
                               GPIOE_BASE }

/** Array initializer of GPIO peripheral base pointers */

#define GPIO_BASE_PTRS          { GPIOA, GPIOB, GPIOC, GPIOD, GPIOE }

```

```

/*
-- GPIO - Register accessor macros
----- */

```

```

/*!
* @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor macros
* @{

```

*/

```
/* GPIO - Register instance definitions */

/* GPIOA */

#define GPIOA_PDOR          GPIO_PDOR_REG(GPIOA)
#define GPIOA_PSOR          GPIO_PSOR_REG(GPIOA)
#define GPIOA_PCOR          GPIO_PCOR_REG(GPIOA)
#define GPIOA_PTOR          GPIO_PTOR_REG(GPIOA)
#define GPIOA_PDIR          GPIO_PDIR_REG(GPIOA)
#define GPIOA_PDDR          GPIO_PDDR_REG(GPIOA)

/* GPIOB */

#define GPIOB_PDOR          GPIO_PDOR_REG(GPIOB)
#define GPIOB_PSOR          GPIO_PSOR_REG(GPIOB)
#define GPIOB_PCOR          GPIO_PCOR_REG(GPIOB)
#define GPIOB_PTOR          GPIO_PTOR_REG(GPIOB)
#define GPIOB_PDIR          GPIO_PDIR_REG(GPIOB)
#define GPIOB_PDDR          GPIO_PDDR_REG(GPIOB)

/* GPIOC */

#define GPIOC_PDOR          GPIO_PDOR_REG(GPIOC)
#define GPIOC_PSOR          GPIO_PSOR_REG(GPIOC)
#define GPIOC_PCOR          GPIO_PCOR_REG(GPIOC)
#define GPIOC_PTOR          GPIO_PTOR_REG(GPIOC)
#define GPIOC_PDIR          GPIO_PDIR_REG(GPIOC)
#define GPIOC_PDDR          GPIO_PDDR_REG(GPIOC)

/* GPIOD */

#define GPIOD_PDOR          GPIO_PDOR_REG(GPIOD)
#define GPIOD_PSOR          GPIO_PSOR_REG(GPIOD)
#define GPIOD_PCOR          GPIO_PCOR_REG(GPIOD)
```

```
#define GPIOD_PTOR          GPIO_PTOR_REG(GPIOD)
#define GPIOD_PDIR          GPIO_PDIR_REG(GPIOD)
#define GPIOD_PDDR          GPIO_PDDR_REG(GPIOD)

/* GPIOE */

#define GPIOE_PDOR          GPIO_PDOR_REG(GPIOE)
#define GPIOE_PSOR          GPIO_PSOR_REG(GPIOE)
#define GPIOE_PCOR          GPIO_PCOR_REG(GPIOE)
#define GPIOE_PTOR          GPIO_PTOR_REG(GPIOE)
#define GPIOE_PDIR          GPIO_PDIR_REG(GPIOE)
#define GPIOE_PDDR          GPIO_PDDR_REG(GPIOE)

/*!

* @}

/* /* end of group GPIO_Register_Accessor_Macros */
```

```
/*!

* @}

/* /* end of group GPIO_Peripheral_Access_Layer */
```

```
/* -----
-- I2C Peripheral Access Layer
----- */
```

```
/*!

* @addtogroup I2C_Peripheral_Access_Layer I2C Peripheral Access Layer
* @{

*/
```

```

/** I2C - Register Layout Typedef */

typedef struct {

    __IO uint8_t A1;           /**< I2C Address Register 1, offset: 0x0 */

    __IO uint8_t F;            /**< I2C Frequency Divider register, offset: 0x1 */

    __IO uint8_t C1;           /**< I2C Control Register 1, offset: 0x2 */

    __IO uint8_t S;            /**< I2C Status register, offset: 0x3 */

    __IO uint8_t D;            /**< I2C Data I/O register, offset: 0x4 */

    __IO uint8_t C2;           /**< I2C Control Register 2, offset: 0x5 */

    __IO uint8_t FLT;          /**< I2C Programmable Input Glitch Filter register, offset: 0x6 */

    __IO uint8_t RA;           /**< I2C Range Address register, offset: 0x7 */

    __IO uint8_t SMB;          /**< I2C SMBus Control and Status register, offset: 0x8 */

    __IO uint8_t A2;           /**< I2C Address Register 2, offset: 0x9 */

    __IO uint8_t SLTH;          /**< I2C SCL Low Timeout Register High, offset: 0xA */

    __IO uint8_t SLTL;          /**< I2C SCL Low Timeout Register Low, offset: 0xB */

} I2C_Type, *I2C_MemMapPtr;

```

```

/*
-----  

-- I2C - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros  

* @{  

*/
```

```

/* I2C - Register accessors */  

#define I2C_A1_REG(base)      ((base)->A1)
```

```
#define I2C_F_REG(base)          ((base)->F)
#define I2C_C1_REG(base)          ((base)->C1)
#define I2C_S_REG(base)           ((base)->S)
#define I2C_D_REG(base)           ((base)->D)
#define I2C_C2_REG(base)          ((base)->C2)
#define I2C_FLT_REG(base)         ((base)->FLT)
#define I2C_RA_REG(base)          ((base)->RA)
#define I2C_SMB_REG(base)         ((base)->SMB)
#define I2C_A2_REG(base)          ((base)->A2)
#define I2C_SLTH_REG(base)        ((base)->SLTH)
#define I2C_SLTL_REG(base)        ((base)->SLTL)
```

```
/*
 * @}
 */
/* /* end of group I2C_Register_Accessor_Macros */
```

```
/*
-----  
-- I2C Register Masks  
----- */
```

```
/*
 * @addtogroup I2C_Register_Masks I2C Register Masks
 * @{
 */

```

```
/* A1 Bit Fields */

#define I2C_A1_AD_MASK           0xFFu
#define I2C_A1_AD_SHIFT          1
```

```

#define I2C_A1_AD_WIDTH          7

#define I2C_A1_AD(x)             (((uint8_t)((uint8_t)(x)<<I2C_A1_AD_SHIFT))&I2C_A1_AD_MASK)

/* F Bit Fields */

#define I2C_F_ICR_MASK           0x3Fu
#define I2C_F_ICR_SHIFT          0
#define I2C_F_ICR_WIDTH          6
#define I2C_F_ICR(x)              (((uint8_t)((uint8_t)(x)<<I2C_F_ICR_SHIFT))&I2C_F_ICR_MASK)

#define I2C_F_MULT_MASK           0xC0u
#define I2C_F_MULT_SHIFT          6
#define I2C_F_MULT_WIDTH          2
#define I2C_F_MULT(x)              (((uint8_t)((uint8_t)(x)<<I2C_F_MULT_SHIFT))&I2C_F_MULT_MASK)

/* C1 Bit Fields */

#define I2C_C1_DMAEN_MASK         0x1u
#define I2C_C1_DMAEN_SHIFT        0
#define I2C_C1_DMAEN_WIDTH        1
#define I2C_C1_DMAEN(x)            (((uint8_t)((uint8_t)(x)<<I2C_C1_DMAEN_SHIFT))&I2C_C1_DMAEN_MASK)

#define I2C_C1_WUEN_MASK           0x2u
#define I2C_C1_WUEN_SHIFT          1
#define I2C_C1_WUEN_WIDTH          1
#define I2C_C1_WUEN(x)              (((uint8_t)((uint8_t)(x)<<I2C_C1_WUEN_SHIFT))&I2C_C1_WUEN_MASK)

#define I2C_C1_RSTA_MASK           0x4u
#define I2C_C1_RSTA_SHIFT          2
#define I2C_C1_RSTA_WIDTH          1
#define I2C_C1_RSTA(x)              (((uint8_t)((uint8_t)(x)<<I2C_C1_RSTA_SHIFT))&I2C_C1_RSTA_MASK)

#define I2C_C1_TXAK_MASK           0x8u

```

```

#define I2C_C1_TXAK_SHIFT          3
#define I2C_C1_TXAK_WIDTH          1
#define I2C_C1_TXAK(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_TXAK_SHIFT))&I2C_C1_TXAK_MASK)

#define I2C_C1_TX_MASK             0x10u
#define I2C_C1_TX_SHIFT            4
#define I2C_C1_TX_WIDTH            1
#define I2C_C1_TX(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_TX_SHIFT))&I2C_C1_TX_MASK)

#define I2C_C1_MST_MASK            0x20u
#define I2C_C1_MST_SHIFT           5
#define I2C_C1_MST_WIDTH           1
#define I2C_C1_MST(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_MST_SHIFT))&I2C_C1_MST_MASK)

#define I2C_C1_IICIE_MASK          0x40u
#define I2C_C1_IICIE_SHIFT          6
#define I2C_C1_IICIE_WIDTH          1
#define I2C_C1_IICIE(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_IICIE_SHIFT))&I2C_C1_IICIE_MASK)

#define I2C_C1_IICEN_MASK          0x80u
#define I2C_C1_IICEN_SHIFT          7
#define I2C_C1_IICEN_WIDTH          1
#define I2C_C1_IICEN(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_IICEN_SHIFT))&I2C_C1_IICEN_MASK)

/* S Bit Fields */

#define I2C_S_RXAK_MASK             0x1u
#define I2C_S_RXAK_SHIFT             0
#define I2C_S_RXAK_WIDTH             1
#define I2C_S_RXAK(x) (((uint8_t)((uint8_t)(x))<<I2C_S_RXAK_SHIFT))&I2C_S_RXAK_MASK)

#define I2C_S_IICIF_MASK            0x2u

```

```

#define I2C_S_IICIF_SHIFT           1
#define I2C_S_IICIF_WIDTH          1
#define I2C_S_IICIF(x)             (((uint8_t)((uint8_t)(x))<<I2C_S_IICIF_SHIFT))&I2C_S_IICIF_MASK)
#define I2C_S_SRW_MASK              0x4u
#define I2C_S_SRW_SHIFT             2
#define I2C_S_SRW_WIDTH             1
#define I2C_S_SRW(x)                (((uint8_t)((uint8_t)(x))<<I2C_S_SRW_SHIFT))&I2C_S_SRW_MASK)
#define I2C_S_RAM_MASK               0x8u
#define I2C_S_RAM_SHIFT              3
#define I2C_S_RAM_WIDTH              1
#define I2C_S_RAM(x)                 (((uint8_t)((uint8_t)(x))<<I2C_S_RAM_SHIFT))&I2C_S_RAM_MASK)
#define I2C_S_ARBL_MASK              0x10u
#define I2C_S_ARBL_SHIFT             4
#define I2C_S_ARBL_WIDTH             1
#define I2C_S_ARBL(x)                (((uint8_t)((uint8_t)(x))<<I2C_S_ARBL_SHIFT))&I2C_S_ARBL_MASK)
#define I2C_S_BUSY_MASK               0x20u
#define I2C_S_BUSY_SHIFT              5
#define I2C_S_BUSY_WIDTH              1
#define I2C_S_BUSY(x)                 (((uint8_t)((uint8_t)(x))<<I2C_S_BUSY_SHIFT))&I2C_S_BUSY_MASK)
#define I2C_S_IAAS_MASK               0x40u
#define I2C_S_IAAS_SHIFT              6
#define I2C_S_IAAS_WIDTH              1
#define I2C_S_IAAS(x)                 (((uint8_t)((uint8_t)(x))<<I2C_S_IAAS_SHIFT))&I2C_S_IAAS_MASK)
#define I2C_S_TCF_MASK                0x80u
#define I2C_S_TCF_SHIFT               7
#define I2C_S_TCF_WIDTH               1

```

```

#define I2C_S_TCF(x)          (((uint8_t)((uint8_t)(x))<<I2C_S_TCF_SHIFT))&I2C_S_TCF_MASK)

/* D Bit Fields */

#define I2C_D_DATA_MASK        0xFFu
#define I2C_D_DATA_SHIFT       0
#define I2C_D_DATA_WIDTH       8

#define I2C_D_DATA(x)          (((uint8_t)((uint8_t)(x))<<I2C_D_DATA_SHIFT))&I2C_D_DATA_MASK)

/* C2 Bit Fields */

#define I2C_C2_AD_MASK         0x7u
#define I2C_C2_AD_SHIFT        0
#define I2C_C2_AD_WIDTH        3

#define I2C_C2_AD(x)           (((uint8_t)((uint8_t)(x))<<I2C_C2_AD_SHIFT))&I2C_C2_AD_MASK)

#define I2C_C2_RMEN_MASK        0x8u
#define I2C_C2_RMEN_SHIFT       3
#define I2C_C2_RMEN_WIDTH       1

#define I2C_C2_RMEN(x)          (((uint8_t)((uint8_t)(x))<<I2C_C2_RMEN_SHIFT))&I2C_C2_RMEN_MASK)

#define I2C_C2_SBRC_MASK        0x10u
#define I2C_C2_SBRC_SHIFT       4
#define I2C_C2_SBRC_WIDTH       1

#define I2C_C2_SBRC(x)          (((uint8_t)((uint8_t)(x))<<I2C_C2_SBRC_SHIFT))&I2C_C2_SBRC_MASK)

#define I2C_C2_HDRS_MASK        0x20u
#define I2C_C2_HDRS_SHIFT       5
#define I2C_C2_HDRS_WIDTH       1

#define I2C_C2_HDRS(x)          (((uint8_t)((uint8_t)(x))<<I2C_C2_HDRS_SHIFT))&I2C_C2_HDRS_MASK)

#define I2C_C2_ADEXT_MASK       0x40u
#define I2C_C2_ADEXT_SHIFT      6
#define I2C_C2_ADEXT_WIDTH      1

```

```

#define I2C_C2_ADEXT(x)
(((uint8_t)((uint8_t)(x)<<I2C_C2_ADEXT_SHIFT))&I2C_C2_ADEXT_MASK)

#define I2C_C2_GCAEN_MASK          0x80u
#define I2C_C2_GCAEN_SHIFT         7
#define I2C_C2_GCAEN_WIDTH         1

#define I2C_C2_GCAEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_C2_GCAEN_SHIFT))&I2C_C2_GCAEN_MASK)

/* FLT Bit Fields */

#define I2C_FLTFLT_MASK           0x1Fu
#define I2C_FLTFLT_SHIFT          0
#define I2C_FLTFLT_WIDTH          5

#define I2C_FLTFLT(x)
(((uint8_t)((uint8_t)(x)<<I2C_FLTFLT_SHIFT))&I2C_FLTFLT_MASK)

#define I2C_FLT_STOPIE_MASK        0x20u
#define I2C_FLT_STOPIE_SHIFT       5
#define I2C_FLT_STOPIE_WIDTH       1

#define I2C_FLT_STOPIE(x)
(((uint8_t)((uint8_t)(x)<<I2C_FLT_STOPIE_SHIFT))&I2C_FLT_STOPIE_MASK)

#define I2C_FLT_STOPF_MASK         0x40u
#define I2C_FLT_STOPF_SHIFT        6
#define I2C_FLT_STOPF_WIDTH        1

#define I2C_FLT_STOPF(x)
(((uint8_t)((uint8_t)(x)<<I2C_FLT_STOPF_SHIFT))&I2C_FLT_STOPF_MASK)

#define I2C_FLT_SHEN_MASK          0x80u
#define I2C_FLT_SHEN_SHIFT         7
#define I2C_FLT_SHEN_WIDTH         1

#define I2C_FLT_SHEN(x)
(((uint8_t)((uint8_t)(x)<<I2C_FLT_SHEN_SHIFT))&I2C_FLT_SHEN_MASK)

/* RA Bit Fields */

#define I2C_RA_RAD_MASK            0xFEu
#define I2C_RA_RAD_SHIFT           1

```

```

#define I2C_RA_RAD_WIDTH          7

#define I2C_RA_RAD(x)
(((uint8_t)((uint8_t)(x)<<I2C_RA_RAD_SHIFT))&I2C_RA_RAD_MASK)

/* SMB Bit Fields */

#define I2C_SMB_SHTF2IE_MASK      0x1u
#define I2C_SMB_SHTF2IE_SHIFT     0
#define I2C_SMB_SHTF2IE_WIDTH     1

#define I2C_SMB_SHTF2IE(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2IE_SHIFT))&I2C_SMB_SHTF2IE_MASK)

#define I2C_SMB_SHTF2_MASK        0x2u
#define I2C_SMB_SHTF2_SHIFT       1
#define I2C_SMB_SHTF2_WIDTH       1

#define I2C_SMB_SHTF2(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF2_SHIFT))&I2C_SMB_SHTF2_MASK)

#define I2C_SMB_SHTF1_MASK        0x4u
#define I2C_SMB_SHTF1_SHIFT       2
#define I2C_SMB_SHTF1_WIDTH       1

#define I2C_SMB_SHTF1(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SHTF1_SHIFT))&I2C_SMB_SHTF1_MASK)

#define I2C_SMB_SLTF_MASK         0x8u
#define I2C_SMB_SLTF_SHIFT        3
#define I2C_SMB_SLTF_WIDTH        1

#define I2C_SMB_SLTF(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_SLTF_SHIFT))&I2C_SMB_SLTF_MASK)

#define I2C_SMB_TCKSEL_MASK       0x10u
#define I2C_SMB_TCKSEL_SHIFT      4
#define I2C_SMB_TCKSEL_WIDTH      1

#define I2C_SMB_TCKSEL(x)
(((uint8_t)((uint8_t)(x)<<I2C_SMB_TCKSEL_SHIFT))&I2C_SMB_TCKSEL_MASK)

#define I2C_SMB_SIICAEN_MASK      0x20u
#define I2C_SMB_SIICAEN_SHIFT     5

```

```

#define I2C_SMB_SIICAEN_WIDTH          1
#define I2C_SMB_SIICAEN(x)             (((uint8_t)((uint8_t)(x)<<I2C_SMB_SIICAEN_SHIFT))&I2C_SMB_SIICAEN_MASK)

#define I2C_SMB_ALERTEN_MASK          0x40u
#define I2C_SMB_ALERTEN_SHIFT         6
#define I2C_SMB_ALERTEN_WIDTH         1
#define I2C_SMB_ALERTEN(x)            (((uint8_t)((uint8_t)(x)<<I2C_SMB_ALERTEN_SHIFT))&I2C_SMB_ALERTEN_MASK)

#define I2C_SMB_FACK_MASK            0x80u
#define I2C_SMB_FACK_SHIFT           7
#define I2C_SMB_FACK_WIDTH           1
#define I2C_SMB_FACK(x)              (((uint8_t)((uint8_t)(x)<<I2C_SMB_FACK_SHIFT))&I2C_SMB_FACK_MASK)

/* A2 Bit Fields */

#define I2C_A2_SAD_MASK              0xFEu
#define I2C_A2_SAD_SHIFT             1
#define I2C_A2_SAD_WIDTH             7
#define I2C_A2_SAD(x)                (((uint8_t)((uint8_t)(x)<<I2C_A2_SAD_SHIFT))&I2C_A2_SAD_MASK)

/* SLTH Bit Fields */

#define I2C_SLTH_SSLT_MASK           0xFFu
#define I2C_SLTH_SSLT_SHIFT          0
#define I2C_SLTH_SSLT_WIDTH          8
#define I2C_SLTH_SSLT(x)             (((uint8_t)((uint8_t)(x)<<I2C_SLTH_SSLT_SHIFT))&I2C_SLTH_SSLT_MASK)

/* SLTL Bit Fields */

#define I2C_SLTL_SSLT_MASK           0xFFu
#define I2C_SLTL_SSLT_SHIFT          0
#define I2C_SLTL_SSLT_WIDTH          8
#define I2C_SLTL_SSLT(x)             (((uint8_t)((uint8_t)(x)<<I2C_SLTL_SSLT_SHIFT))&I2C_SLTL_SSLT_MASK)

```

```

/*!

* @}

/* /* end of group I2C_Register_Masks */

/* I2C - Peripheral instance base addresses */

/** Peripheral I2C0 base address */

#define I2C0_BASE          (0x40066000u)

/** Peripheral I2C0 base pointer */

#define I2C0          ((I2C_Type *)I2C0_BASE)

#define I2C0_BASE_PTR      (I2C0)

/** Peripheral I2C1 base address */

#define I2C1_BASE          (0x40067000u)

/** Peripheral I2C1 base pointer */

#define I2C1          ((I2C_Type *)I2C1_BASE)

#define I2C1_BASE_PTR      (I2C1)

/** Array initializer of I2C peripheral base addresses */

#define I2C_BASE_ADDRS      { I2C0_BASE, I2C1_BASE }

/** Array initializer of I2C peripheral base pointers */

#define I2C_BASE_PTRS       { I2C0, I2C1 }

/*
-----  

-- I2C - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor macros  

* @{

```

*/

```
/* I2C - Register instance definitions */

/* I2C0 */

#define I2C0_A1          I2C_A1_REG(I2C0)
#define I2C0_F           I2C_F_REG(I2C0)
#define I2C0_C1          I2C_C1_REG(I2C0)
#define I2C0_S           I2C_S_REG(I2C0)
#define I2C0_D           I2C_D_REG(I2C0)
#define I2C0_C2          I2C_C2_REG(I2C0)
#define I2C0_FLT         I2C_FLT_REG(I2C0)
#define I2C0_RA          I2C_RA_REG(I2C0)
#define I2C0_SMB         I2C_SMB_REG(I2C0)
#define I2C0_A2          I2C_A2_REG(I2C0)
#define I2C0_SLTH        I2C_SLTH_REG(I2C0)
#define I2C0_SLTL        I2C_SLTL_REG(I2C0)

/* I2C1 */

#define I2C1_A1          I2C_A1_REG(I2C1)
#define I2C1_F           I2C_F_REG(I2C1)
#define I2C1_C1          I2C_C1_REG(I2C1)
#define I2C1_S           I2C_S_REG(I2C1)
#define I2C1_D           I2C_D_REG(I2C1)
#define I2C1_C2          I2C_C2_REG(I2C1)
#define I2C1_FLT         I2C_FLT_REG(I2C1)
#define I2C1_RA          I2C_RA_REG(I2C1)
#define I2C1_SMB         I2C_SMB_REG(I2C1)
#define I2C1_A2          I2C_A2_REG(I2C1)
#define I2C1_SLTH        I2C_SLTH_REG(I2C1)
```

```

#define I2C1_SLTL           I2C_SLTL_REG(I2C1)

/*
 * @}
 */

/* /* end of group I2C_Register_Accessor_Macros */

/*
 * @}
 */

/* /* end of group I2C_Peripheral_Access_Layer */

/*
-----*
-- LLWU Peripheral Access Layer
----- */
/*!

* @addtogroup LLWU_Peripheral_Access_Layer LLWU Peripheral Access Layer
* @{
*/
/* ** LLWU - Register Layout Typedef */

typedef struct {

    __IO uint8_t PE1;          /**< LLWU Pin Enable 1 register, offset: 0x0 */
    __IO uint8_t PE2;          /**< LLWU Pin Enable 2 register, offset: 0x1 */
    __IO uint8_t PE3;          /**< LLWU Pin Enable 3 register, offset: 0x2 */
    __IO uint8_t PE4;          /**< LLWU Pin Enable 4 register, offset: 0x3 */
    __IO uint8_t ME;           /**< LLWU Module Enable register, offset: 0x4 */
    __IO uint8_t F1;           /**< LLWU Flag 1 register, offset: 0x5 */
}

```

```

__IO uint8_t F2;           /**< LLWU Flag 2 register, offset: 0x6 */
__I uint8_t F3;            /**< LLWU Flag 3 register, offset: 0x7 */
__IO uint8_t FILT1;        /**< LLWU Pin Filter 1 register, offset: 0x8 */
__IO uint8_t FILT2;        /**< LLWU Pin Filter 2 register, offset: 0x9 */

} LLWU_Type, *LLWU_MemMapPtr;

```

```

/*
-----*
-- LLWU - Register accessor macros
----- */

```

```

/*!
* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros
* @{
*/

```

```

/* LLWU - Register accessors */

#define LLWU_PE1_REG(base)      ((base)->PE1)
#define LLWU_PE2_REG(base)      ((base)->PE2)
#define LLWU_PE3_REG(base)      ((base)->PE3)
#define LLWU_PE4_REG(base)      ((base)->PE4)
#define LLWU_ME_REG(base)       ((base)->ME)
#define LLWU_F1_REG(base)       ((base)->F1)
#define LLWU_F2_REG(base)       ((base)->F2)
#define LLWU_F3_REG(base)       ((base)->F3)
#define LLWU_FILT1_REG(base)    ((base)->FILT1)
#define LLWU_FILT2_REG(base)    ((base)->FILT2)

```

```

/*!
```

```
* @}
*/
/* end of group LLWU_Register_Accessor_Macros */
```

```
/*
-----*
-- LLWU Register Masks
----- */
```

```
/*!
* @addtogroup LLWU_Register_Masks LLWU Register Masks
* @{
*/

```

```
/* PE1 Bit Fields */

#define LLWU_PE1_WUPE0_MASK          0x3u
#define LLWU_PE1_WUPE0_SHIFT         0
#define LLWU_PE1_WUPE0_WIDTH         2
#define LLWU_PE1_WUPE0(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE0_SHIFT))&LLWU_PE1_WUPE0_MASK)

#define LLWU_PE1_WUPE1_MASK          0xCu
#define LLWU_PE1_WUPE1_SHIFT         2
#define LLWU_PE1_WUPE1_WIDTH         2
#define LLWU_PE1_WUPE1(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE1_SHIFT))&LLWU_PE1_WUPE1_MASK)

#define LLWU_PE1_WUPE2_MASK          0x30u
#define LLWU_PE1_WUPE2_SHIFT         4
#define LLWU_PE1_WUPE2_WIDTH         2
#define LLWU_PE1_WUPE2(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE2_SHIFT))&LLWU_PE1_WUPE2_MASK)

#define LLWU_PE1_WUPE3_MASK          0xC0u
```

```

#define LLWU_PE1_WUPE3_SHIFT          6
#define LLWU_PE1_WUPE3_WIDTH          2
#define LLWU_PE1_WUPE3(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE1_WUPE3_SHIFT))&LLWU_PE1_WUPE3_MASK)

/* PE2 Bit Fields */

#define LLWU_PE2_WUPE4_MASK           0x3u
#define LLWU_PE2_WUPE4_SHIFT          0
#define LLWU_PE2_WUPE4_WIDTH          2
#define LLWU_PE2_WUPE4(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE4_SHIFT))&LLWU_PE2_WUPE4_MASK)

#define LLWU_PE2_WUPE5_MASK           0xCu
#define LLWU_PE2_WUPE5_SHIFT          2
#define LLWU_PE2_WUPE5_WIDTH          2
#define LLWU_PE2_WUPE5(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE5_SHIFT))&LLWU_PE2_WUPE5_MASK)

#define LLWU_PE2_WUPE6_MASK           0x30u
#define LLWU_PE2_WUPE6_SHIFT          4
#define LLWU_PE2_WUPE6_WIDTH          2
#define LLWU_PE2_WUPE6(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE6_SHIFT))&LLWU_PE2_WUPE6_MASK)

#define LLWU_PE2_WUPE7_MASK           0xC0u
#define LLWU_PE2_WUPE7_SHIFT          6
#define LLWU_PE2_WUPE7_WIDTH          2
#define LLWU_PE2_WUPE7(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE2_WUPE7_SHIFT))&LLWU_PE2_WUPE7_MASK)

/* PE3 Bit Fields */

#define LLWU_PE3_WUPE8_MASK           0x3u
#define LLWU_PE3_WUPE8_SHIFT          0
#define LLWU_PE3_WUPE8_WIDTH          2
#define LLWU_PE3_WUPE8(x) (((uint8_t)((uint8_t)(x)<<LLWU_PE3_WUPE8_SHIFT))&LLWU_PE3_WUPE8_MASK)

```

```

#define LLWU_PE3_WUPE9_MASK          0xCu
#define LLWU_PE3_WUPE9_SHIFT         2
#define LLWU_PE3_WUPE9_WIDTH         2
#define LLWU_PE3_WUPE9(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE9_SHIFT))&LLWU_PE3_WUPE9_MASK)

#define LLWU_PE3_WUPE10_MASK         0x30u
#define LLWU_PE3_WUPE10_SHIFT        4
#define LLWU_PE3_WUPE10_WIDTH        2
#define LLWU_PE3_WUPE10(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE10_SHIFT))&LLWU_PE3_WUPE10_MASK)

#define LLWU_PE3_WUPE11_MASK         0xC0u
#define LLWU_PE3_WUPE11_SHIFT        6
#define LLWU_PE3_WUPE11_WIDTH        2
#define LLWU_PE3_WUPE11(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE11_SHIFT))&LLWU_PE3_WUPE11_MASK)

/* PE4 Bit Fields */

#define LLWU_PE4_WUPE12_MASK         0x3u
#define LLWU_PE4_WUPE12_SHIFT        0
#define LLWU_PE4_WUPE12_WIDTH        2
#define LLWU_PE4_WUPE12(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE12_SHIFT))&LLWU_PE4_WUPE12_MASK)

#define LLWU_PE4_WUPE13_MASK         0xCu
#define LLWU_PE4_WUPE13_SHIFT        2
#define LLWU_PE4_WUPE13_WIDTH        2
#define LLWU_PE4_WUPE13(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE13_SHIFT))&LLWU_PE4_WUPE13_MASK)

#define LLWU_PE4_WUPE14_MASK         0x30u
#define LLWU_PE4_WUPE14_SHIFT        4
#define LLWU_PE4_WUPE14_WIDTH        2
#define LLWU_PE4_WUPE14(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE14_SHIFT))&LLWU_PE4_WUPE14_MASK)

```

```

#define LLWU_PE4_WUPE15_MASK          0xC0u
#define LLWU_PE4_WUPE15_SHIFT         6
#define LLWU_PE4_WUPE15_WIDTH         2
#define LLWU_PE4_WUPE15(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE15_SHIFT))&LLWU_PE4_WUPE15_MASK)

/* ME Bit Fields */

#define LLWU_ME_WUME0_MASK           0x1u
#define LLWU_ME_WUME0_SHIFT          0
#define LLWU_ME_WUME0_WIDTH          1
#define LLWU_ME_WUME0(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME0_SHIFT))&LLWU_ME_WUME0_MASK)

#define LLWU_ME_WUME1_MASK           0x2u
#define LLWU_ME_WUME1_SHIFT          1
#define LLWU_ME_WUME1_WIDTH          1
#define LLWU_ME_WUME1(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME1_SHIFT))&LLWU_ME_WUME1_MASK)

#define LLWU_ME_WUME2_MASK           0x4u
#define LLWU_ME_WUME2_SHIFT          2
#define LLWU_ME_WUME2_WIDTH          1
#define LLWU_ME_WUME2(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME2_SHIFT))&LLWU_ME_WUME2_MASK)

#define LLWU_ME_WUME3_MASK           0x8u
#define LLWU_ME_WUME3_SHIFT          3
#define LLWU_ME_WUME3_WIDTH          1
#define LLWU_ME_WUME3(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME3_SHIFT))&LLWU_ME_WUME3_MASK)

#define LLWU_ME_WUME4_MASK           0x10u
#define LLWU_ME_WUME4_SHIFT          4
#define LLWU_ME_WUME4_WIDTH          1
#define LLWU_ME_WUME4(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME4_SHIFT))&LLWU_ME_WUME4_MASK)

```

```

#define LLWU_ME_WUME5_MASK          0x20u
#define LLWU_ME_WUME5_SHIFT         5
#define LLWU_ME_WUME5_WIDTH          1
#define LLWU_ME_WUME5(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME5_SHIFT))&LLWU_ME_WUME5_MASK)

#define LLWU_ME_WUME6_MASK          0x40u
#define LLWU_ME_WUME6_SHIFT         6
#define LLWU_ME_WUME6_WIDTH          1
#define LLWU_ME_WUME6(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME6_SHIFT))&LLWU_ME_WUME6_MASK)

#define LLWU_ME_WUME7_MASK          0x80u
#define LLWU_ME_WUME7_SHIFT         7
#define LLWU_ME_WUME7_WIDTH          1
#define LLWU_ME_WUME7(x) (((uint8_t)((uint8_t)(x))<<LLWU_ME_WUME7_SHIFT))&LLWU_ME_WUME7_MASK)

/* F1 Bit Fields */

#define LLWU_F1_WUFO_MASK           0x1u
#define LLWU_F1_WUFO_SHIFT          0
#define LLWU_F1_WUFO_WIDTH          1
#define LLWU_F1_WUFO(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUFO_SHIFT))&LLWU_F1_WUFO_MASK)

#define LLWU_F1_WUF1_MASK           0x2u
#define LLWU_F1_WUF1_SHIFT          1
#define LLWU_F1_WUF1_WIDTH          1
#define LLWU_F1_WUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF1_SHIFT))&LLWU_F1_WUF1_MASK)

#define LLWU_F1_WUF2_MASK           0x4u
#define LLWU_F1_WUF2_SHIFT          2
#define LLWU_F1_WUF2_WIDTH          1
#define LLWU_F1_WUF2(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF2_SHIFT))&LLWU_F1_WUF2_MASK)

```

```

#define LLWU_F1_WUF3_MASK          0x8u
#define LLWU_F1_WUF3_SHIFT         3
#define LLWU_F1_WUF3_WIDTH         1
#define LLWU_F1_WUF3(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF3_SHIFT))&LLWU_F1_WUF3_MASK)

#define LLWU_F1_WUF4_MASK          0x10u
#define LLWU_F1_WUF4_SHIFT         4
#define LLWU_F1_WUF4_WIDTH         1
#define LLWU_F1_WUF4(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF4_SHIFT))&LLWU_F1_WUF4_MASK)

#define LLWU_F1_WUF5_MASK          0x20u
#define LLWU_F1_WUF5_SHIFT         5
#define LLWU_F1_WUF5_WIDTH         1
#define LLWU_F1_WUF5(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF5_SHIFT))&LLWU_F1_WUF5_MASK)

#define LLWU_F1_WUF6_MASK          0x40u
#define LLWU_F1_WUF6_SHIFT         6
#define LLWU_F1_WUF6_WIDTH         1
#define LLWU_F1_WUF6(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF6_SHIFT))&LLWU_F1_WUF6_MASK)

#define LLWU_F1_WUF7_MASK          0x80u
#define LLWU_F1_WUF7_SHIFT         7
#define LLWU_F1_WUF7_WIDTH         1
#define LLWU_F1_WUF7(x) (((uint8_t)((uint8_t)(x)<<LLWU_F1_WUF7_SHIFT))&LLWU_F1_WUF7_MASK)

/* F2 Bit Fields */

#define LLWU_F2_WUF8_MASK          0x1u
#define LLWU_F2_WUF8_SHIFT         0
#define LLWU_F2_WUF8_WIDTH         1
#define LLWU_F2_WUF8(x) (((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF8_SHIFT))&LLWU_F2_WUF8_MASK)

```

```
#define LLWU_F2_WUF9_MASK          0x2u
#define LLWU_F2_WUF9_SHIFT          1
#define LLWU_F2_WUF9_WIDTH          1
#define LLWU_F2_WUF9(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF9_SHIFT))&LLWU_F2_WUF9_MASK)

#define LLWU_F2_WUF10_MASK          0x4u
#define LLWU_F2_WUF10_SHIFT          2
#define LLWU_F2_WUF10_WIDTH          1
#define LLWU_F2_WUF10(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF10_SHIFT))&LLWU_F2_WUF10_MASK)

#define LLWU_F2_WUF11_MASK          0x8u
#define LLWU_F2_WUF11_SHIFT          3
#define LLWU_F2_WUF11_WIDTH          1
#define LLWU_F2_WUF11(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF11_SHIFT))&LLWU_F2_WUF11_MASK)

#define LLWU_F2_WUF12_MASK          0x10u
#define LLWU_F2_WUF12_SHIFT          4
#define LLWU_F2_WUF12_WIDTH          1
#define LLWU_F2_WUF12(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF12_SHIFT))&LLWU_F2_WUF12_MASK)

#define LLWU_F2_WUF13_MASK          0x20u
#define LLWU_F2_WUF13_SHIFT          5
#define LLWU_F2_WUF13_WIDTH          1
#define LLWU_F2_WUF13(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF13_SHIFT))&LLWU_F2_WUF13_MASK)

#define LLWU_F2_WUF14_MASK          0x40u
#define LLWU_F2_WUF14_SHIFT          6
#define LLWU_F2_WUF14_WIDTH          1
#define LLWU_F2_WUF14(x)
(((uint8_t)((uint8_t)(x)<<LLWU_F2_WUF14_SHIFT))&LLWU_F2_WUF14_MASK)

#define LLWU_F2_WUF15_MASK          0x80u
```

```

#define LLWU_F2_WUF15_SHIFT          7
#define LLWU_F2_WUF15_WIDTH          1
#define LLWU_F2_WUF15(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF15_SHIFT))&LLWU_F2_WUF15_MASK)

/* F3 Bit Fields */

#define LLWU_F3_MWUF0_MASK           0x1u
#define LLWU_F3_MWUF0_SHIFT          0
#define LLWU_F3_MWUF0_WIDTH          1
#define LLWU_F3_MWUF0(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF0_SHIFT))&LLWU_F3_MWUF0_MASK)

#define LLWU_F3_MWUF1_MASK           0x2u
#define LLWU_F3_MWUF1_SHIFT          1
#define LLWU_F3_MWUF1_WIDTH          1
#define LLWU_F3_MWUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF1_SHIFT))&LLWU_F3_MWUF1_MASK)

#define LLWU_F3_MWUF2_MASK           0x4u
#define LLWU_F3_MWUF2_SHIFT          2
#define LLWU_F3_MWUF2_WIDTH          1
#define LLWU_F3_MWUF2(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF2_SHIFT))&LLWU_F3_MWUF2_MASK)

#define LLWU_F3_MWUF3_MASK           0x8u
#define LLWU_F3_MWUF3_SHIFT          3
#define LLWU_F3_MWUF3_WIDTH          1
#define LLWU_F3_MWUF3(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF3_SHIFT))&LLWU_F3_MWUF3_MASK)

#define LLWU_F3_MWUF4_MASK           0x10u
#define LLWU_F3_MWUF4_SHIFT          4
#define LLWU_F3_MWUF4_WIDTH          1
#define LLWU_F3_MWUF4(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF4_SHIFT))&LLWU_F3_MWUF4_MASK)

#define LLWU_F3_MWUF5_MASK           0x20u

```

```

#define LLWU_F3_MWUF5_SHIFT          5
#define LLWU_F3_MWUF5_WIDTH          1
#define LLWU_F3_MWUF5(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF5_SHIFT))&LLWU_F3_MWUF5_MASK)

#define LLWU_F3_MWUF6_MASK           0x40u
#define LLWU_F3_MWUF6_SHIFT          6
#define LLWU_F3_MWUF6_WIDTH          1
#define LLWU_F3_MWUF6(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF6_SHIFT))&LLWU_F3_MWUF6_MASK)

#define LLWU_F3_MWUF7_MASK           0x80u
#define LLWU_F3_MWUF7_SHIFT          7
#define LLWU_F3_MWUF7_WIDTH          1
#define LLWU_F3_MWUF7(x) (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF7_SHIFT))&LLWU_F3_MWUF7_MASK)

/* FILT1 Bit Fields */

#define LLWU_FILT1_FILTSEL_MASK      0xFu
#define LLWU_FILT1_FILTSEL_SHIFT     0
#define LLWU_FILT1_FILTSEL_WIDTH     4
#define LLWU_FILT1_FILTSEL(x) (((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTSEL_SHIFT))&LLWU_FILT1_FILTSEL_MASK)

#define LLWU_FILT1_FILTE_MASK        0x60u
#define LLWU_FILT1_FILTE_SHIFT       5
#define LLWU_FILT1_FILTE_WIDTH       2
#define LLWU_FILT1_FILTE(x) (((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTE_SHIFT))&LLWU_FILT1_FILTE_MASK)

#define LLWU_FILT1_FILTF_MASK        0x80u
#define LLWU_FILT1_FILTF_SHIFT       7
#define LLWU_FILT1_FILTF_WIDTH       1
#define LLWU_FILT1_FILTF(x) (((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTF_SHIFT))&LLWU_FILT1_FILTF_MASK)

/* FILT2 Bit Fields */

```

```

#define LLWU_FILT2_FILTSEL_MASK      0xFu
#define LLWU_FILT2_FILTSEL_SHIFT     0
#define LLWU_FILT2_FILTSEL_WIDTH     4
#define LLWU_FILT2_FILTSEL(x)        (((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTSEL_SHIFT))&LLWU_FILT2_FILTSEL_MASK)
#define LLWU_FILT2_FILTE_MASK        0x60u
#define LLWU_FILT2_FILTE_SHIFT       5
#define LLWU_FILT2_FILTE_WIDTH       2
#define LLWU_FILT2_FILTE(x)          (((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTE_SHIFT))&LLWU_FILT2_FILTE_MASK)
#define LLWU_FILT2_FILTF_MASK        0x80u
#define LLWU_FILT2_FILTF_SHIFT       7
#define LLWU_FILT2_FILTF_WIDTH       1
#define LLWU_FILT2_FILTF(x)          (((uint8_t)((uint8_t)(x)<<LLWU_FILT2_FILTF_SHIFT))&LLWU_FILT2_FILTF_MASK)

```

```

/*
 * @}
 */
/* end of group LLWU_Register_Masks */

```

```

/* LLWU - Peripheral instance base addresses */
/** Peripheral LLWU base address */
#define LLWU_BASE                  (0x4007C000u)
/** Peripheral LLWU base pointer */
#define LLWU                      ((LLWU_Type *)LLWU_BASE)
#define LLWU_BASE_PTR              (LLWU)
/** Array initializer of LLWU peripheral base addresses */
#define LLWU_BASE_ADDRS            { LLWU_BASE }
/** Array initializer of LLWU peripheral base pointers */

```

```

#define LLWU_BASE_PTRS           { LLWU }

/*
-----  

-- LLWU - Register accessor macros  

----- */
/*!  

* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor macros  

* @{
* /  

  

/* LLWU - Register instance definitions */  

/* LLWU */  

#define LLWU_PE1                 LLWU_PE1_REG(LLWU)  

#define LLWU_PE2                 LLWU_PE2_REG(LLWU)  

#define LLWU_PE3                 LLWU_PE3_REG(LLWU)  

#define LLWU_PE4                 LLWU_PE4_REG(LLWU)  

#define LLWU_ME                  LLWU_ME_REG(LLWU)  

#define LLWU_F1                  LLWU_F1_REG(LLWU)  

#define LLWU_F2                  LLWU_F2_REG(LLWU)  

#define LLWU_F3                  LLWU_F3_REG(LLWU)  

#define LLWU_FILT1               LLWU_FILT1_REG(LLWU)  

#define LLWU_FILT2               LLWU_FILT2_REG(LLWU)  

  

/*!  

* @}
* /* end of group LLWU_Register_Accessor_Macros */

```

```

/*!
 * @}
 */

/* /* end of group LLWU_Peripheral_Access_Layer */

/*
-----*
-- LPTMR Peripheral Access Layer
----- */
/*!

 * @addtogroup LPTMR_Peripheral_Access_Layer LPTMR Peripheral Access Layer
 * @{
 */

/** LPTMR - Register Layout Typedef */

typedef struct {

    __IO uint32_t CSR;           /**< Low Power Timer Control Status Register, offset: 0x0 */
    __IO uint32_t PSR;           /**< Low Power Timer Prescale Register, offset: 0x4 */
    __IO uint32_t CMR;           /**< Low Power Timer Compare Register, offset: 0x8 */
    __IO uint32_t CNR;           /**< Low Power Timer Counter Register, offset: 0xC */

} LPTMR_Type, *LPTMR_MemMapPtr;

/*
-----*
-- LPTMR - Register accessor macros
----- */
/*!

 * @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
 */

```

```
* @{

/*
 * LPTMR - Register accessors */

#define LPTMR_CSR_REG(base)      ((base)->CSR)
#define LPTMR_PSR_REG(base)      ((base)->PSR)
#define LPTMR_CMR_REG(base)      ((base)->CMR)
#define LPTMR_CNR_REG(base)      ((base)->CNR)

/*!

 * @}

 */ /* end of group LPTMR_Register_Accessor_Macros */

```

```
/*
-----  
-- LPTMR Register Masks  
-----*/

```

```
/*!  
 * @addtogroup LPTMR_Register_Masks LPTMR Register Masks  
 * @{
 * /
```

```
/* CSR Bit Fields */

#define LPTMR_CSR_TEN_MASK      0x1u
#define LPTMR_CSR_TEN_SHIFT     0
#define LPTMR_CSR_TEN_WIDTH     1
```

```

#define LPTMR_CSR_TEN(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR_TEN_SHIFT))&LPTMR_CSR_TEN_MASK)

#define LPTMR_CSR_TMS_MASK          0x2u
#define LPTMR_CSR_TMS_SHIFT         1
#define LPTMR_CSR_TMS_WIDTH         1

#define LPTMR_CSR_TMS(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR_TMS_SHIFT))&LPTMR_CSR_TMS_MASK)

#define LPTMR_CSR_TFC_MASK          0x4u
#define LPTMR_CSR_TFC_SHIFT         2
#define LPTMR_CSR_TFC_WIDTH         1

#define LPTMR_CSR_TFC(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR_TFC_SHIFT))&LPTMR_CSR_TFC_MASK)

#define LPTMR_CSR TPP_MASK          0x8u
#define LPTMR_CSR TPP_SHIFT         3
#define LPTMR_CSR TPP_WIDTH         1

#define LPTMR_CSR TPP(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR TPP_SHIFT))&LPTMR_CSR TPP_MASK)

#define LPTMR_CSR TPS_MASK          0x30u
#define LPTMR_CSR TPS_SHIFT         4
#define LPTMR_CSR TPS_WIDTH         2

#define LPTMR_CSR TPS(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR TPS_SHIFT))&LPTMR_CSR TPS_MASK)

#define LPTMR_CSR TIE_MASK          0x40u
#define LPTMR_CSR TIE_SHIFT         6
#define LPTMR_CSR TIE_WIDTH         1

#define LPTMR_CSR TIE(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR TIE_SHIFT))&LPTMR_CSR TIE_MASK)

#define LPTMR_CSR TCF_MASK          0x80u
#define LPTMR_CSR TCF_SHIFT         7
#define LPTMR_CSR TCF_WIDTH         1

```

```

#define LPTMR_CSR_TCF(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CSR_TCF_SHIFT))&LPTMR_CSR_TCF_MASK)

/* PSR Bit Fields */

#define LPTMR_PSR_PCS_MASK          0x3u
#define LPTMR_PSR_PCS_SHIFT         0
#define LPTMR_PSR_PCS_WIDTH         2

#define LPTMR_PSR_PCS(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_PSR_PCS_SHIFT))&LPTMR_PSR_PCS_MASK)

#define LPTMR_PSR_PBYP_MASK         0x4u
#define LPTMR_PSR_PBYP_SHIFT        2
#define LPTMR_PSR_PBYP_WIDTH        1

#define LPTMR_PSR_PBYP(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_PSR_PBYP_SHIFT))&LPTMR_PSR_PBYP_MASK)

#define LPTMR_PSR_PRESCALE_MASK     0x78u
#define LPTMR_PSR_PRESCALE_SHIFT    3
#define LPTMR_PSR_PRESCALE_WIDTH    4

#define LPTMR_PSR_PRESCALE(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_PSR_PRESCALE_SHIFT))&LPTMR_PSR_PRESCALE_MASK)

/* CMR Bit Fields */

#define LPTMR_CMR_COMPARE_MASK      0xFFFFu
#define LPTMR_CMR_COMPARE_SHIFT     0
#define LPTMR_CMR_COMPARE_WIDTH     16

#define LPTMR_CMR_COMPARE(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CMR_COMPARE_SHIFT))&LPTMR_CMR_COMPARE_MASK)

/* CNR Bit Fields */

#define LPTMR_CNR_COUNTER_MASK      0xFFFFu
#define LPTMR_CNR_COUNTER_SHIFT     0
#define LPTMR_CNR_COUNTER_WIDTH     16

#define LPTMR_CNR_COUNTER(x)
(((uint32_t)((uint32_t)(x)<<LPTMR_CNR_COUNTER_SHIFT))&LPTMR_CNR_COUNTER_MASK)

```

```

/*!
* @}
*/
/* /* end of group LPTMR_Register_Masks */

/* LPTMR - Peripheral instance base addresses */
/** Peripheral LPTMRO base address */
#define LPTMRO_BASE           (0x40040000u)
/** Peripheral LPTMRO base pointer */
#define LPTMRO                 ((LPTMR_Type *)LPTMRO_BASE)
#define LPTMRO_BASE_PTR        (LPTMRO)
/** Array initializer of LPTMR peripheral base addresses */
#define LPTMR_BASE_ADDRS       { LPTMRO_BASE }
/** Array initializer of LPTMR peripheral base pointers */
#define LPTMR_BASE_PTRS        { LPTMRO }

/*
-- LPTMR - Register accessor macros
----- */
/*!
* @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
* @{
*/
/* LPTMR - Register instance definitions */
/* LPTMRO */
#define LPTMRO_CSR             LPTMR_CSR_REG(LPTMRO)

```

```
#define LPTMRO_PSR           LPTMR_PSR_REG(LPTMRO)
#define LPTMRO_CMR           LPTMR_CMREG(LPTMRO)
#define LPTMRO_CNR           LPTMR_CNR_REG(LPTMRO)

/*!
 * @}
 */

/* /* end of group LPTMR_Register_Accessor_Macros */
```

```
/*!
 * @}
 */

/* /* end of group LPTMR_Peripheral_Access_Layer */
```

```
/*
-----  
-- MCG Peripheral Access Layer  
----- */
```

```
/*!  
* @addtogroup MCG_Peripheral_Access_Layer MCG Peripheral Access Layer  
* @{  
*/
```

```
/** MCG - Register Layout Typedef */  
  
typedef struct {  
    __IO uint8_t C1;          /**< MCG Control 1 Register, offset: 0x0 */  
    __IO uint8_t C2;          /**< MCG Control 2 Register, offset: 0x1 */  
    __IO uint8_t C3;          /**< MCG Control 3 Register, offset: 0x2 */  
    __IO uint8_t C4;          /**< MCG Control 4 Register, offset: 0x3 */
```

```

__IO uint8_t C5;           /**< MCG Control 5 Register, offset: 0x4 */
__IO uint8_t C6;           /**< MCG Control 6 Register, offset: 0x5 */
__IO uint8_t S;            /**< MCG Status Register, offset: 0x6 */
uint8_t RESERVED_0[1];
__IO uint8_t SC;           /**< MCG Status and Control Register, offset: 0x8 */
uint8_t RESERVED_1[1];
__IO uint8_t ATCVH;        /**< MCG Auto Trim Compare Value High Register, offset: 0xA */
__IO uint8_t ATCVL;        /**< MCG Auto Trim Compare Value Low Register, offset: 0xB */
__I uint8_t C7;            /**< MCG Control 7 Register, offset: 0xC */
__IO uint8_t C8;            /**< MCG Control 8 Register, offset: 0xD */
__I uint8_t C9;            /**< MCG Control 9 Register, offset: 0xE */
__I uint8_t C10;           /**< MCG Control 10 Register, offset: 0xF */

} MCG_Type, *MCG_MemMapPtr;

```

```

/*
-----*
-- MCG - Register accessor macros
----- */

```

```

/*!
* @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros
* @{
*/

```

```

/* MCG - Register accessors */

#define MCG_C1_REG(base)      ((base)->C1)
#define MCG_C2_REG(base)      ((base)->C2)
#define MCG_C3_REG(base)      ((base)->C3)
#define MCG_C4_REG(base)      ((base)->C4)

```

```
#define MCG_C5_REG(base) ((base)->C5)
#define MCG_C6_REG(base) ((base)->C6)
#define MCG_S_REG(base) ((base)->S)
#define MCG_SC_REG(base) ((base)->SC)
#define MCG_ATCVH_REG(base) ((base)->ATCVH)
#define MCG_ATCVL_REG(base) ((base)->ATCVL)
#define MCG_C7_REG(base) ((base)->C7)
#define MCG_C8_REG(base) ((base)->C8)
#define MCG_C9_REG(base) ((base)->C9)
#define MCG_C10_REG(base) ((base)->C10)
```

```
/*
 * @}
 */
/* /* end of group MCG_Register_Accessor_Macros */
```

```
/*
-----  
-- MCG Register Masks  
----- */
```

```
/*
 * @addtogroup MCG_Register_Masks MCG Register Masks
 * @{
 */

```

```
/* C1 Bit Fields */

#define MCG_C1_IREFSTEN_MASK 0x1u
#define MCG_C1_IREFSTEN_SHIFT 0
#define MCG_C1_IREFSTEN_WIDTH 1
```

```

#define MCG_C1_IREFSTEN(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_IREFSTEN_SHIFT))&MCG_C1_IREFSTEN_MASK)

#define MCG_C1_IRCLKEN_MASK          0x2u
#define MCG_C1_IRCLKEN_SHIFT         1
#define MCG_C1_IRCLKEN_WIDTH         1

#define MCG_C1_IRCLKEN(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_IRCLKEN_SHIFT))&MCG_C1_IRCLKEN_MASK)

#define MCG_C1_IREFS_MASK            0x4u
#define MCG_C1_IREFS_SHIFT           2
#define MCG_C1_IREFS_WIDTH           1

#define MCG_C1_IREFS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_IREFS_SHIFT))&MCG_C1_IREFS_MASK)

#define MCG_C1_FRDIV_MASK            0x38u
#define MCG_C1_FRDIV_SHIFT           3
#define MCG_C1_FRDIV_WIDTH           3

#define MCG_C1_FRDIV(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_FRDIV_SHIFT))&MCG_C1_FRDIV_MASK)

#define MCG_C1_CLKS_MASK              0xC0u
#define MCG_C1_CLKS_SHIFT             6
#define MCG_C1_CLKS_WIDTH             2

#define MCG_C1_CLKS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C1_CLKS_SHIFT))&MCG_C1_CLKS_MASK)

/* C2 Bit Fields */

#define MCG_C2_IRCS_MASK              0x1u
#define MCG_C2_IRCS_SHIFT              0
#define MCG_C2_IRCS_WIDTH              1

#define MCG_C2_IRCS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_IRCS_SHIFT))&MCG_C2_IRCS_MASK)

#define MCG_C2_LP_MASK                0x2u
#define MCG_C2_LP_SHIFT                 1
#define MCG_C2_LP_WIDTH                  1

```

```

#define MCG_C2_LP(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_LP_SHIFT))&MCG_C2_LP_MASK)

#define MCG_C2_EREF0_MASK          0x4u
#define MCG_C2_EREF0_SHIFT         2
#define MCG_C2_EREF0_WIDTH         1

#define MCG_C2_EREF0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_EREF0_SHIFT))&MCG_C2_EREF0_MASK)

#define MCG_C2_HGO0_MASK          0x8u
#define MCG_C2_HGO0_SHIFT         3
#define MCG_C2_HGO0_WIDTH         1

#define MCG_C2_HGO0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_HGO0_SHIFT))&MCG_C2_HGO0_MASK)

#define MCG_C2_RANGE0_MASK         0x30u
#define MCG_C2_RANGE0_SHIFT        4
#define MCG_C2_RANGE0_WIDTH        2

#define MCG_C2_RANGE0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_RANGE0_SHIFT))&MCG_C2_RANGE0_MASK)

#define MCG_C2_LOCRE0_MASK         0x80u
#define MCG_C2_LOCRE0_SHIFT        7
#define MCG_C2_LOCRE0_WIDTH        1

#define MCG_C2_LOCRE0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C2_LOCRE0_SHIFT))&MCG_C2_LOCRE0_MASK)

/* C3 Bit Fields */

#define MCG_C3_SCTRIM_MASK         0xFFu
#define MCG_C3_SCTRIM_SHIFT        0
#define MCG_C3_SCTRIM_WIDTH        8

#define MCG_C3_SCTRIM(x)
(((uint8_t)((uint8_t)(x)<<MCG_C3_SCTRIM_SHIFT))&MCG_C3_SCTRIM_MASK)

/* C4 Bit Fields */

#define MCG_C4_SCFTRIM_MASK         0x1u
#define MCG_C4_SCFTRIM_SHIFT        0

```

```

#define MCG_C4_SCFTRIM_WIDTH          1
#define MCG_C4_SCFTRIM(x)             (((uint8_t)((uint8_t)(x)<<MCG_C4_SCFTRIM_SHIFT))&MCG_C4_SCFTRIM_MASK)
#define MCG_C4_FCTRIM_MASK           0x1Eu
#define MCG_C4_FCTRIM_SHIFT          1
#define MCG_C4_FCTRIM_WIDTH          4
#define MCG_C4_FCTRIM(x)             (((uint8_t)((uint8_t)(x)<<MCG_C4_FCTRIM_SHIFT))&MCG_C4_FCTRIM_MASK)
#define MCG_C4_DRST_DRS_MASK         0x60u
#define MCG_C4_DRST_DRS_SHIFT        5
#define MCG_C4_DRST_DRS_WIDTH        2
#define MCG_C4_DRST_DRS(x)           (((uint8_t)((uint8_t)(x)<<MCG_C4_DRST_DRS_SHIFT))&MCG_C4_DRST_DRS_MASK)
#define MCG_C4_DMX32_MASK            0x80u
#define MCG_C4_DMX32_SHIFT           7
#define MCG_C4_DMX32_WIDTH           1
#define MCG_C4_DMX32(x)              (((uint8_t)((uint8_t)(x)<<MCG_C4_DMX32_SHIFT))&MCG_C4_DMX32_MASK)

/* C5 Bit Fields */

#define MCG_C5_PRDIV0_MASK           0x1Fu
#define MCG_C5_PRDIV0_SHIFT          0
#define MCG_C5_PRDIV0_WIDTH          5
#define MCG_C5_PRDIV0(x)             (((uint8_t)((uint8_t)(x)<<MCG_C5_PRDIV0_SHIFT))&MCG_C5_PRDIV0_MASK)
#define MCG_C5_PLLSTENO_MASK          0x20u
#define MCG_C5_PLLSTENO_SHIFT         5
#define MCG_C5_PLLSTENO_WIDTH        1
#define MCG_C5_PLLSTENO(x)           (((uint8_t)((uint8_t)(x)<<MCG_C5_PLLSTENO_SHIFT))&MCG_C5_PLLSTENO_MASK)
#define MCG_C5_PLLCLKENO_MASK         0x40u
#define MCG_C5_PLLCLKENO_SHIFT        6

```

```

#define MCG_C5_PLLCLKENO_WIDTH          1
#define MCG_C5_PLLCLKENO(x)
(((uint8_t)((uint8_t)(x)<<MCG_C5_PLLCLKENO_SHIFT))&MCG_C5_PLLCLKENO_MASK)

/* C6 Bit Fields */

#define MCG_C6_VDIV0_MASK              0x1Fu
#define MCG_C6_VDIV0_SHIFT             0
#define MCG_C6_VDIV0_WIDTH             5
#define MCG_C6_VDIV0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_VDIV0_SHIFT))&MCG_C6_VDIV0_MASK)

#define MCG_C6_CME0_MASK               0x20u
#define MCG_C6_CME0_SHIFT              5
#define MCG_C6_CME0_WIDTH              1
#define MCG_C6_CME0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_CME0_SHIFT))&MCG_C6_CME0_MASK)

#define MCG_C6_PLLS_MASK               0x40u
#define MCG_C6_PLLS_SHIFT              6
#define MCG_C6_PLLS_WIDTH              1
#define MCG_C6_PLLS(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_PLLS_SHIFT))&MCG_C6_PLLS_MASK)

#define MCG_C6_LOLIE0_MASK              0x80u
#define MCG_C6_LOLIE0_SHIFT             7
#define MCG_C6_LOLIE0_WIDTH             1
#define MCG_C6_LOLIE0(x)
(((uint8_t)((uint8_t)(x)<<MCG_C6_LOLIE0_SHIFT))&MCG_C6_LOLIE0_MASK)

/* S Bit Fields */

#define MCG_S_IRCST_MASK                0x1u
#define MCG_S_IRCST_SHIFT               0
#define MCG_S_IRCST_WIDTH               1
#define MCG_S_IRCST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_IRCST_SHIFT))&MCG_S_IRCST_MASK)

#define MCG_S_OSCINIT0_MASK              0x2u

```

```

#define MCG_S_OSCINIT0_SHIFT          1
#define MCG_S_OSCINIT0_WIDTH          1
#define MCG_S_OSCINIT0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_OSCINIT0_SHIFT))&MCG_S_OSCINIT0_MASK)

#define MCG_S_CLKST_MASK              0xCu
#define MCG_S_CLKST_SHIFT             2
#define MCG_S_CLKST_WIDTH             2
#define MCG_S_CLKST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_CLKST_SHIFT))&MCG_S_CLKST_MASK)

#define MCG_S_IREFST_MASK             0x10u
#define MCG_S_IREFST_SHIFT            4
#define MCG_S_IREFST_WIDTH            1
#define MCG_S_IREFST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_IREFST_SHIFT))&MCG_S_IREFST_MASK)

#define MCG_S_PLLST_MASK              0x20u
#define MCG_S_PLLST_SHIFT             5
#define MCG_S_PLLST_WIDTH             1
#define MCG_S_PLLST(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_PLLST_SHIFT))&MCG_S_PLLST_MASK)

#define MCG_S_LOCK0_MASK               0x40u
#define MCG_S_LOCK0_SHIFT              6
#define MCG_S_LOCK0_WIDTH              1
#define MCG_S_LOCK0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_LOCK0_SHIFT))&MCG_S_LOCK0_MASK)

#define MCG_S_LOLS0_MASK               0x80u
#define MCG_S_LOLS0_SHIFT              7
#define MCG_S_LOLS0_WIDTH              1
#define MCG_S_LOLS0(x)
(((uint8_t)((uint8_t)(x)<<MCG_S_LOLS0_SHIFT))&MCG_S_LOLS0_MASK)

/* SC Bit Fields */

#define MCG_SC_LOCS0_MASK              0x1u

```

```

#define MCG_SC_LOCS0_SHIFT          0
#define MCG_SC_LOCS0_WIDTH          1
#define MCG_SC_LOCS0(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_LOCS0_SHIFT))&MCG_SC_LOCS0_MASK)

#define MCG_SC_FCRDIV_MASK          0xEu
#define MCG_SC_FCRDIV_SHIFT         1
#define MCG_SC_FCRDIV_WIDTH         3
#define MCG_SC_FCRDIV(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_FCRDIV_SHIFT))&MCG_SC_FCRDIV_MASK)

#define MCG_SC_FLTPRSRV_MASK         0x10u
#define MCG_SC_FLTPRSRV_SHIFT        4
#define MCG_SC_FLTPRSRV_WIDTH        1
#define MCG_SC_FLTPRSRV(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_FLTPRSRV_SHIFT))&MCG_SC_FLTPRSRV_MASK)

#define MCG_SC_ATMF_MASK             0x20u
#define MCG_SC_ATMF_SHIFT            5
#define MCG_SC_ATMF_WIDTH            1
#define MCG_SC_ATMF(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMF_SHIFT))&MCG_SC_ATMF_MASK)

#define MCG_SC_ATMS_MASK             0x40u
#define MCG_SC_ATMS_SHIFT            6
#define MCG_SC_ATMS_WIDTH            1
#define MCG_SC_ATMS(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATMS_SHIFT))&MCG_SC_ATMS_MASK)

#define MCG_SC_ATME_MASK             0x80u
#define MCG_SC_ATME_SHIFT            7
#define MCG_SC_ATME_WIDTH            1
#define MCG_SC_ATME(x) (((uint8_t)((uint8_t)(x)<<MCG_SC_ATME_SHIFT))&MCG_SC_ATME_MASK)

/* ATCVH Bit Fields */

#define MCG_ATCVH_ATCVH_MASK         0xFFu

```

```

#define MCG_ATCVH_ATCVH_SHIFT          0
#define MCG_ATCVH_ATCVH_WIDTH          8
#define MCG_ATCVH_ATCVH(x)
(((uint8_t)((uint8_t)(x)<<MCG_ATCVH_ATCVH_SHIFT))&MCG_ATCVH_ATCVH_MASK)

/* ATCVL Bit Fields */

#define MCG_ATCVL_ATCVL_MASK           0xFFu
#define MCG_ATCVL_ATCVL_SHIFT          0
#define MCG_ATCVL_ATCVL_WIDTH          8
#define MCG_ATCVL_ATCVL(x)
(((uint8_t)((uint8_t)(x)<<MCG_ATCVL_ATCVL_SHIFT))&MCG_ATCVL_ATCVL_MASK)

/* C8 Bit Fields */

#define MCG_C8_LOLRE_MASK              0x40u
#define MCG_C8_LOLRE_SHIFT             6
#define MCG_C8_LOLRE_WIDTH             1
#define MCG_C8_LOLRE(x)
(((uint8_t)((uint8_t)(x)<<MCG_C8_LOLRE_SHIFT))&MCG_C8_LOLRE_MASK)

/*!
 * @}
 */

/* /* end of group MCG_Register_Masks */

```

```

/* MCG - Peripheral instance base addresses */

/** Peripheral MCG base address */

#define MCG_BASE                      (0x40064000u)

/** Peripheral MCG base pointer */

#define MCG                         ((MCG_Type *)MCG_BASE)

#define MCG_BASE_PTR                  (MCG)

/** Array initializer of MCG peripheral base addresses */

#define MCG_BASE_ADDRS                { MCG_BASE }

```

```

/** Array initializer of MCG peripheral base pointers */

#define MCG_BASE_PTRS { MCG }

/*
-- MCG - Register accessor macros
----- */
/*!
* @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor macros
* @{
*/
/* MCG - Register instance definitions */

/* MCG */

#define MCG_C1 MCG_C1_REG(MCG)
#define MCG_C2 MCG_C2_REG(MCG)
#define MCG_C3 MCG_C3_REG(MCG)
#define MCG_C4 MCG_C4_REG(MCG)
#define MCG_C5 MCG_C5_REG(MCG)
#define MCG_C6 MCG_C6_REG(MCG)
#define MCG_S MCG_S_REG(MCG)
#define MCG_SC MCG_SC_REG(MCG)
#define MCG_ATCVH MCG_ATCVH_REG(MCG)
#define MCG_ATCVL MCG_ATCVL_REG(MCG)
#define MCG_C7 MCG_C7_REG(MCG)
#define MCG_C8 MCG_C8_REG(MCG)
#define MCG_C9 MCG_C9_REG(MCG)
#define MCG_C10 MCG_C10_REG(MCG)

```

```
/*!  
 * @}  
 */ /* end of group MCG_Register_Accessor_Macros */  
  
/* MCG C2[EREFS] backward compatibility */  
  
#define MCG_C2_EREFs_MASK      (MCG_C2_EREFs0_MASK)  
#define MCG_C2_EREFs_SHIFT     (MCG_C2_EREFs0_SHIFT)  
#define MCG_C2_EREFs_WIDTH     (MCG_C2_EREFs0_WIDTH)  
#define MCG_C2_EREFs(x)        (MCG_C2_EREFs0(x))  
  
/* MCG C2[HGO] backward compatibility */  
  
#define MCG_C2_HGO_MASK       (MCG_C2_HGO0_MASK)  
#define MCG_C2_HGO_SHIFT      (MCG_C2_HGO0_SHIFT)  
#define MCG_C2_HGO_WIDTH      (MCG_C2_HGO0_WIDTH)  
#define MCG_C2_HGO(x)         (MCG_C2_HGO0(x))  
  
/* MCG C2[RANGE] backward compatibility */  
  
#define MCG_C2_RANGE_MASK     (MCG_C2_RANGE0_MASK)  
#define MCG_C2_RANGE_SHIFT    (MCG_C2_RANGE0_SHIFT)  
#define MCG_C2_RANGE_WIDTH    (MCG_C2_RANGE0_WIDTH)  
#define MCG_C2_RANGE(x)       (MCG_C2_RANGE0(x))  
  
/*!  
 * @}  
 */ /* end of group MCG_Peripheral_Access_Layer */
```

```

/*
-----  

-- MCM Peripheral Access Layer  

----- */
  

  

/*!  

 * @addtogroup MCM_Peripheral_Access_Layer MCM Peripheral Access Layer
 * @{
 * /  

  

/** MCM - Register Layout Typedef */  

typedef struct {  

    uint8_t RESERVED_0[8];  

    __I uint16_t PLASC;           /**< Crossbar Switch (AXBS) Slave Configuration, offset: 0x8 */  

    __I uint16_t PLAMC;           /**< Crossbar Switch (AXBS) Master Configuration, offset: 0xA */  

    __IO uint32_t PLACR;          /**< Platform Control Register, offset: 0xC */  

    uint8_t RESERVED_1[48];  

    __IO uint32_t CPO;            /**< Compute Operation Control Register, offset: 0x40 */  

} MCM_Type, *MCM_MemMapPtr;  

  

/*
-----  

-- MCM - Register accessor macros  

----- */
  

  

/*!  

 * @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros
 * @{
 * /
```

```

/* MCM - Register accessors */

#define MCM_PLASC_REG(base)          ((base)->PLASC)
#define MCM_PLAMC_REG(base)          ((base)->PLAMC)
#define MCM_PLACR_REG(base)          ((base)->PLACR)
#define MCM_CPO_REG(base)           ((base)->CPO)

/*
 * @}
 */
/* /* end of group MCM_Register_Accessor_Macros */

```

```

/*
-----  

-- MCM Register Masks  

----- */

```

```

/*
* @addtogroup MCM_Register_Masks MCM Register Masks
* @{
*/

```

```

/* PLASC Bit Fields */

#define MCM_PLASC_ASC_MASK          0xFFu
#define MCM_PLASC_ASC_SHIFT         0
#define MCM_PLASC_ASC_WIDTH         8

#define MCM_PLASC_ASC(x)            (((uint16_t)((uint16_t)(x)<<MCM_PLASC_ASC_SHIFT))&MCM_PLASC_ASC_MASK)

/* PLAMC Bit Fields */

#define MCM_PLAMC_AMC_MASK          0xFFu
#define MCM_PLAMC_AMC_SHIFT         0

```

```

#define MCM_PLAMC_AMC_WIDTH          8
#define MCM_PLAMC_AMC(x) (((uint16_t)((uint16_t)(x)<<MCM_PLAMC_AMC_SHIFT))&MCM_PLAMC_AMC_MASK)
/* PLACR Bit Fields */
#define MCM_PLACR_ARB_MASK           0x200u
#define MCM_PLACR_ARB_SHIFT          9
#define MCM_PLACR_ARB_WIDTH          1
#define MCM_PLACR_ARB(x) (((uint32_t)((uint32_t)(x)<<MCM_PLACR_ARB_SHIFT))&MCM_PLACR_ARB_MASK)
#define MCM_PLACR_CFCC_MASK          0x400u
#define MCM_PLACR_CFCC_SHIFT         10
#define MCM_PLACR_CFCC_WIDTH         1
#define MCM_PLACR_CFCC(x) (((uint32_t)((uint32_t)(x)<<MCM_PLACR_CFCC_SHIFT))&MCM_PLACR_CFCC_MASK)
#define MCM_PLACR_DFCDA_MASK         0x800u
#define MCM_PLACR_DFCDA_SHIFT        11
#define MCM_PLACR_DFCDA_WIDTH        1
#define MCM_PLACR_DFCDA(x) (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCDA_SHIFT))&MCM_PLACR_DFCDA_MASK)
#define MCM_PLACR_DFCIC_MASK         0x1000u
#define MCM_PLACR_DFCIC_SHIFT        12
#define MCM_PLACR_DFCIC_WIDTH        1
#define MCM_PLACR_DFCIC(x) (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCIC_SHIFT))&MCM_PLACR_DFCIC_MASK)
#define MCM_PLACR_DFCC_MASK          0x2000u
#define MCM_PLACR_DFCC_SHIFT         13
#define MCM_PLACR_DFCC_WIDTH         1
#define MCM_PLACR_DFCC(x) (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCC_SHIFT))&MCM_PLACR_DFCC_MASK)
#define MCM_PLACR_EFDS_MASK          0x4000u
#define MCM_PLACR_EFDS_SHIFT         14

```

```

#define MCM_PLACR_EFDS_WIDTH          1
#define MCM_PLACR_EFDS(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_EFDS_SHIFT))&MCM_PLACR_EFDS_MASK)
#define MCM_PLACR_DFCS_MASK           0x8000u
#define MCM_PLACR_DFCS_SHIFT          15
#define MCM_PLACR_DFCS_WIDTH          1
#define MCM_PLACR_DFCS(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_DFCS_SHIFT))&MCM_PLACR_DFCS_MASK)
#define MCM_PLACR_ESFC_MASK           0x10000u
#define MCM_PLACR_ESFC_SHIFT          16
#define MCM_PLACR_ESFC_WIDTH          1
#define MCM_PLACR_ESFC(x)             (((uint32_t)((uint32_t)(x)<<MCM_PLACR_ESFC_SHIFT))&MCM_PLACR_ESFC_MASK)

/* CPO Bit Fields */

#define MCM_CPO_CPOREQ_MASK           0x1u
#define MCM_CPO_CPOREQ_SHIFT          0
#define MCM_CPO_CPOREQ_WIDTH          1
#define MCM_CPO_CPOREQ(x)             (((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOREQ_SHIFT))&MCM_CPO_CPOREQ_MASK)
#define MCM_CPO_CPOACK_MASK           0x2u
#define MCM_CPO_CPOACK_SHIFT          1
#define MCM_CPO_CPOACK_WIDTH          1
#define MCM_CPO_CPOACK(x)             (((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOACK_SHIFT))&MCM_CPO_CPOACK_MASK)
#define MCM_CPO_CPOWOI_MASK           0x4u
#define MCM_CPO_CPOWOI_SHIFT          2
#define MCM_CPO_CPOWOI_WIDTH          1
#define MCM_CPO_CPOWOI(x)             (((uint32_t)((uint32_t)(x)<<MCM_CPO_CPOWOI_SHIFT))&MCM_CPO_CPOWOI_MASK)

*/

```

```

* @}

/* /* end of group MCM_Register_Masks */

/* MCM - Peripheral instance base addresses */

/** Peripheral MCM base address */

#define MCM_BASE          (0xF0003000u)

/** Peripheral MCM base pointer */

#define MCM      ((MCM_Type *)MCM_BASE)

#define MCM_BASE_PTR      (MCM)

/** Array initializer of MCM peripheral base addresses */

#define MCM_BASE_ADDRS    { MCM_BASE }

/** Array initializer of MCM peripheral base pointers */

#define MCM_BASE_PTRS     { MCM }

/*
-----  

-- MCM - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor macros  

* @{
* /  

  

/* MCM - Register instance definitions */  

/* MCM */  

#define MCM_PLASC          MCM_PLASC_REG(MCM)  

#define MCM_PLAMC          MCM_PLAMC_REG(MCM)

```

```

#define MCM_PLACR           MCM_PLACR_REG(MCM)
#define MCM_CPO              MCM_CPO_REG(MCM)

/*!
* @}

/* /* end of group MCM_Register_Accessor_Macros */

/*!
* @}

/* /* end of group MCM_Peripheral_Access_Layer */

/*
-----  

-- MTB Peripheral Access Layer  

----- */
/*!

* @addtogroup MTB_Peripheral_Access_Layer MTB Peripheral Access Layer
* @{
* /


/** MTB - Register Layout Typedef */

typedef struct {
    __IO uint32_t POSITION;          /**< MTB Position Register, offset: 0x0 */
    __IO uint32_t MASTER;            /**< MTB Master Register, offset: 0x4 */
    __IO uint32_t FLOW;              /**< MTB Flow Register, offset: 0x8 */
    __I  uint32_t BASE;              /**< MTB Base Register, offset: 0xC */
    uint8_t RESERVED_0[3824];
}

```

```

__I uint32_t MODECTRL;           /**< Integration Mode Control Register, offset: 0xFO0 */
uint8_t RESERVED_1[156];

__I uint32_t TAGSET;            /**< Claim TAG Set Register, offset: 0xFA0 */
__I uint32_t TAGCLEAR;          /**< Claim TAG Clear Register, offset: 0xFA4 */

uint8_t RESERVED_2[8];

__I uint32_t LOCKACCESS;         /**< Lock Access Register, offset: 0xFB0 */
__I uint32_t LOCKSTAT;          /**< Lock Status Register, offset: 0xFB4 */
__I uint32_t AUTHSTAT;          /**< Authentication Status Register, offset: 0xFB8 */
__I uint32_t DEVICEARCH;        /**< Device Architecture Register, offset: 0xFBC */

uint8_t RESERVED_3[8];

__I uint32_t DEVICECFG;         /**< Device Configuration Register, offset: 0xFC8 */
__I uint32_t DEVICETYPID;       /**< Device Type Identifier Register, offset: 0xFCC */
__I uint32_t PERIPHID[8];       /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4 */

__I uint32_t COMPID[4];         /**< Component ID Register, array offset: 0xFF0, array step: 0x4 */
*/
}

} MTB_Type, *MTB_MemMapPtr;

```

```

/*
-----*
-- MTB - Register accessor macros
----- */

```

```

/*!
* @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros
* @{
*/

```

```

/* MTB - Register accessors */

```

```

#define MTB_POSITION_REG(base)          ((base)->POSITION)
#define MTB_MASTER_REG(base)           ((base)->MASTER)
#define MTB_FLOW_REG(base)             ((base)->FLOW)
#define MTB_BASE_REG(base)             ((base)->BASE)
#define MTB_MODECTRL_REG(base)         ((base)->MODECTRL)
#define MTB_TAGSET_REG(base)           ((base)->TAGSET)
#define MTB_TAGCLEAR_REG(base)         ((base)->TAGCLEAR)
#define MTB_LOCKACCESS_REG(base)       ((base)->LOCKACCESS)
#define MTB_LOCKSTAT_REG(base)         ((base)->LOCKSTAT)
#define MTB_AUTHSTAT_REG(base)         ((base)->AUTHSTAT)
#define MTB_DEVICEARCH_REG(base)       ((base)->DEVICEARCH)
#define MTB_DEVICECFG_REG(base)        ((base)->DEVICECFG)
#define MTB_DEVICETYPID_REG(base)      ((base)->DEVICETYPID)
#define MTB_PERIPHID_REG(base,index)   ((base)->PERIPHID[index])
#define MTB_PERIPHID_COUNT            8
#define MTB_COMPID_REG(base,index)     ((base)->COMPID[index])
#define MTB_COMPID_COUNT              4

```

```

/*
 * @}
 */
 /* end of group MTB_Register_Accessor_Macros */

```

```

/*
-----*
-- MTB Register Masks
----- */

```

```

/*
 * @addtogroup MTB_Register_Masks MTB Register Masks

```

```

* @{

*/



/* POSITION Bit Fields */

#define MTB_POSITION_WRAP_MASK          0x4u
#define MTB_POSITION_WRAP_SHIFT        2
#define MTB_POSITION_WRAP_WIDTH        1

#define MTB_POSITION_WRAP(x)
(((uint32_t)((uint32_t)(x))<<MTB_POSITION_WRAP_SHIFT))&MTB_POSITION_WRAP_MASK)

#define MTB_POSITION_POINTER_MASK      0xFFFFFFFF8u
#define MTB_POSITION_POINTER_SHIFT    3
#define MTB_POSITION_POINTER_WIDTH    29

#define MTB_POSITION_POINTER(x)
(((uint32_t)((uint32_t)(x))<<MTB_POSITION_POINTER_SHIFT))&MTB_POSITION_POINTER_MASK

/* MASTER Bit Fields */

#define MTB_MASTER_MASK_MASK          0x1Fu
#define MTB_MASTER_MASK_SHIFT         0
#define MTB_MASTER_MASK_WIDTH         5

#define MTB_MASTER_MASK(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_MASK_SHIFT))&MTB_MASTER_MASK_MASK)

#define MTB_MASTER_TSTARTEN_MASK      0x20u
#define MTB_MASTER_TSTARTEN_SHIFT     5
#define MTB_MASTER_TSTARTEN_WIDTH     1

#define MTB_MASTER_TSTARTEN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTARTEN_SHIFT))&MTB_MASTER_TSTARTEN_MASK)

#define MTB_MASTER_TSTOPEN_MASK       0x40u
#define MTB_MASTER_TSTOPEN_SHIFT      6
#define MTB_MASTER_TSTOPEN_WIDTH      1

#define MTB_MASTER_TSTOPEN(x)
(((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTOPEN_SHIFT))&MTB_MASTER_TSTOPEN_MASK)

```

```

#define MTB_MASTER_SFRWPRIV_MASK      0x80u
#define MTB_MASTER_SFRWPRIV_SHIFT     7
#define MTB_MASTER_SFRWPRIV_WIDTH     1
#define MTB_MASTER_SFRWPRIV(x) (((uint32_t)((uint32_t)(x))<<MTB_MASTER_SFRWPRIV_SHIFT))&MTB_MASTER_SFRWPRIV_MASK)

#define MTB_MASTER_RAMPRIV_MASK       0x100u
#define MTB_MASTER_RAMPRIV_SHIFT      8
#define MTB_MASTER_RAMPRIV_WIDTH      1
#define MTB_MASTER_RAMPRIV(x) (((uint32_t)((uint32_t)(x))<<MTB_MASTER_RAMPRIV_SHIFT))&MTB_MASTER_RAMPRIV_MASK)

#define MTB_MASTER_HALTREQ_MASK       0x200u
#define MTB_MASTER_HALTREQ_SHIFT      9
#define MTB_MASTER_HALTREQ_WIDTH      1
#define MTB_MASTER_HALTREQ(x) (((uint32_t)((uint32_t)(x))<<MTB_MASTER_HALTREQ_SHIFT))&MTB_MASTER_HALTREQ_MASK)

#define MTB_MASTER_EN_MASK            0x80000000u
#define MTB_MASTER_EN_SHIFT           31
#define MTB_MASTER_EN_WIDTH           1
#define MTB_MASTER_EN(x) (((uint32_t)((uint32_t)(x))<<MTB_MASTER_EN_SHIFT))&MTB_MASTER_EN_MASK)

/* FLOW Bit Fields */

#define MTB_FLOW_AUTOSTOP_MASK        0x1u
#define MTB_FLOW_AUTOSTOP_SHIFT       0
#define MTB_FLOW_AUTOSTOP_WIDTH       1
#define MTB_FLOW_AUTOSTOP(x) (((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOSTOP_SHIFT))&MTB_FLOW_AUTOSTOP_MASK)

#define MTB_FLOW_AUTOHALT_MASK         0x2u
#define MTB_FLOW_AUTOHALT_SHIFT        1
#define MTB_FLOW_AUTOHALT_WIDTH        1
#define MTB_FLOW_AUTOHALT(x) (((uint32_t)((uint32_t)(x))<<MTB_FLOW_AUTOHALT_SHIFT))&MTB_FLOW_AUTOHALT_MASK)

```

```

#define MTB_FLOW_WATERMARK_MASK      0xFFFFFFFF8u
#define MTB_FLOW_WATERMARK_SHIFT     3
#define MTB_FLOW_WATERMARK_WIDTH    29
#define MTB_FLOW_WATERMARK(x)        (((uint32_t)((uint32_t)(x))<<MTB_FLOW_WATERMARK_SHIFT))&MTB_FLOW_WATERMARK_MASK)

/* BASE Bit Fields */

#define MTB_BASE_BASEADDR_MASK       0xFFFFFFFFFu
#define MTB_BASE_BASEADDR_SHIFT      0
#define MTB_BASE_BASEADDR_WIDTH     32
#define MTB_BASE_BASEADDR(x)         (((uint32_t)((uint32_t)(x))<<MTB_BASE_BASEADDR_SHIFT))&MTB_BASE_BASEADDR_MASK)

/* MODECTRL Bit Fields */

#define MTB_MODECTRL_MODECTRL_MASK   0xFFFFFFFFFu
#define MTB_MODECTRL_MODECTRL_SHIFT  0
#define MTB_MODECTRL_MODECTRL_WIDTH 32
#define MTB_MODECTRL_MODECTRL(x)     (((uint32_t)((uint32_t)(x))<<MTB_MODECTRL_MODECTRL_SHIFT))&MTB_MODECTRL_MODECTRL_MASK)

/* TAGSET Bit Fields */

#define MTB_TAGSET_TAGSET_MASK       0xFFFFFFFFFu
#define MTB_TAGSET_TAGSET_SHIFT      0
#define MTB_TAGSET_TAGSET_WIDTH     32
#define MTB_TAGSET_TAGSET(x)         (((uint32_t)((uint32_t)(x))<<MTB_TAGSET_TAGSET_SHIFT))&MTB_TAGSET_TAGSET_MASK)

/* TAGCLEAR Bit Fields */

#define MTB_TAGCLEAR_TAGCLEAR_MASK   0xFFFFFFFFFu
#define MTB_TAGCLEAR_TAGCLEAR_SHIFT  0
#define MTB_TAGCLEAR_TAGCLEAR_WIDTH 32
#define MTB_TAGCLEAR_TAGCLEAR(x)     (((uint32_t)((uint32_t)(x))<<MTB_TAGCLEAR_TAGCLEAR_SHIFT))&MTB_TAGCLEAR_TAGCLEAR_MASK)

/* LOCKACCESS Bit Fields */

```

```

#define MTB_LOCKACCESS_LOCKACCESS_MASK      0xFFFFFFFFu
#define MTB_LOCKACCESS_LOCKACCESS_SHIFT     0
#define MTB_LOCKACCESS_LOCKACCESS_WIDTH     32
#define MTB_LOCKACCESS_LOCKACCESS(x)        (((uint32_t)((uint32_t)(x)<<MTB_LOCKACCESS_LOCKACCESS_SHIFT))&MTB_LOCKACCESS_LOCKACCESS_MASK)

/* LOCKSTAT Bit Fields */

#define MTB_LOCKSTAT_LOCKSTAT_MASK         0xFFFFFFFFu
#define MTB_LOCKSTAT_LOCKSTAT_SHIFT       0
#define MTB_LOCKSTAT_LOCKSTAT_WIDTH       32
#define MTB_LOCKSTAT_LOCKSTAT(x)          (((uint32_t)((uint32_t)(x)<<MTB_LOCKSTAT_LOCKSTAT_SHIFT))&MTB_LOCKSTAT_LOCKSTAT_MASK)

/* AUTHSTAT Bit Fields */

#define MTB_AUTHSTAT_BIT0_MASK            0x1u
#define MTB_AUTHSTAT_BIT0_SHIFT          0
#define MTB_AUTHSTAT_BIT0_WIDTH          1
#define MTB_AUTHSTAT_BIT0(x)             (((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT0_SHIFT))&MTB_AUTHSTAT_BIT0_MASK)

#define MTB_AUTHSTAT_BIT1_MASK           0x2u
#define MTB_AUTHSTAT_BIT1_SHIFT          1
#define MTB_AUTHSTAT_BIT1_WIDTH          1
#define MTB_AUTHSTAT_BIT1(x)             (((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT1_SHIFT))&MTB_AUTHSTAT_BIT1_MASK)

#define MTB_AUTHSTAT_BIT2_MASK           0x4u
#define MTB_AUTHSTAT_BIT2_SHIFT          2
#define MTB_AUTHSTAT_BIT2_WIDTH          1
#define MTB_AUTHSTAT_BIT2(x)             (((uint32_t)((uint32_t)(x)<<MTB_AUTHSTAT_BIT2_SHIFT))&MTB_AUTHSTAT_BIT2_MASK)

#define MTB_AUTHSTAT_BIT3_MASK           0x8u
#define MTB_AUTHSTAT_BIT3_SHIFT          3
#define MTB_AUTHSTAT_BIT3_WIDTH          1

```

```

#define MTB_AUTHSTAT_BIT3(x)
(((uint32_t)((uint32_t)(x))<<MTB_AUTHSTAT_BIT3_SHIFT))&MTB_AUTHSTAT_BIT3_MASK)

/* DEVICEARCH Bit Fields */

#define MTB_DEVICEARCH_DEVICEARCH_MASK      0xFFFFFFFFu
#define MTB_DEVICEARCH_DEVICEARCH_SHIFT     0
#define MTB_DEVICEARCH_DEVICEARCH_WIDTH     32

#define MTB_DEVICEARCH_DEVICEARCH(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICEARCH_DEVICEARCH_SHIFT))&MTB_DEVICEARCH_DEVICEARCH_MASK

/* DEVICECFG Bit Fields */

#define MTB_DEVICECFG_DEVICECFG_MASK      0xFFFFFFFFu
#define MTB_DEVICECFG_DEVICECFG_SHIFT     0
#define MTB_DEVICECFG_DEVICECFG_WIDTH     32

#define MTB_DEVICECFG_DEVICECFG(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICECFG_DEVICECFG_SHIFT))&MTB_DEVICECFG_DEVICECFG_MASK

/* DEVICETYPID Bit Fields */

#define MTB_DEVICETYPID_DEVICETYPID_MASK   0xFFFFFFFFu
#define MTB_DEVICETYPID_DEVICETYPID_SHIFT  0
#define MTB_DEVICETYPID_DEVICETYPID_WIDTH  32

#define MTB_DEVICETYPID_DEVICETYPID(x)
(((uint32_t)((uint32_t)(x))<<MTB_DEVICETYPID_DEVICETYPID_SHIFT))&MTB_DEVICETYPID_DEVICETYPID_MASK

/* PERIPHID Bit Fields */

#define MTB_PERIPHID_PERIPHID_MASK      0xFFFFFFFFu
#define MTB_PERIPHID_PERIPHID_SHIFT     0
#define MTB_PERIPHID_PERIPHID_WIDTH    32

#define MTB_PERIPHID_PERIPHID(x)
(((uint32_t)((uint32_t)(x))<<MTB_PERIPHID_PERIPHID_SHIFT))&MTB_PERIPHID_PERIPHID_MASK

/* COMPID Bit Fields */

#define MTB_COMPID_COMPID_MASK      0xFFFFFFFFu
#define MTB_COMPID_COMPID_SHIFT     0

```

```

#define MTB_COMPID_COMPID_WIDTH      32

#define MTB_COMPID_COMPID(x)
((uint32_t)((uint32_t)(x)<<MTB_COMPID_COMPID_SHIFT))&MTB_COMPID_COMPID_MASK)

/*
 * @}
 */

/* /* end of group MTB_Register_Masks */

/* MTB - Peripheral instance base addresses */
/** Peripheral MTB base address */
#define MTB_BASE          (0xF0000000u)

/** Peripheral MTB base pointer */
#define MTB          ((MTB_Type *)MTB_BASE)

#define MTB_BASE_PTR      (MTB)

/** Array initializer of MTB peripheral base addresses */
#define MTB_BASE_ADDRS    { MTB_BASE }

/** Array initializer of MTB peripheral base pointers */
#define MTB_BASE_PTRS     { MTB }

/*
----- --
-- MTB - Register accessor macros
----- */

```

```

/*
 * @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor macros
 * @{
 */


```

```
/* MTB - Register instance definitions */

/* MTB */

#define MTB_POSITION          MTB_POSITION_REG(MTB)
#define MTB_MASTER             MTB_MASTER_REG(MTB)
#define MTB_FLOW               MTB_FLOW_REG(MTB)
#define MTB_BASEr              MTB_BASE_REG(MTB)
#define MTB_MODECTRL           MTB_MODECTRL_REG(MTB)
#define MTB_TAGSET              MTB_TAGSET_REG(MTB)
#define MTB_TAGCLEAR            MTB_TAGCLEAR_REG(MTB)
#define MTB_LOCKACCESS           MTB_LOCKACCESS_REG(MTB)
#define MTB_LOCKSTAT             MTB_LOCKSTAT_REG(MTB)
#define MTB_AUTHSTAT             MTB_AUTHSTAT_REG(MTB)
#define MTB_DEVICESEARCH          MTB_DEVICESEARCH_REG(MTB)
#define MTB_DEVICECFG             MTB_DEVICECFG_REG(MTB)
#define MTB_DEVICETYPID          MTB_DEVICETYPID_REG(MTB)
#define MTB_PERIPHID4            MTB_PERIPHID_REG(MTB,0)
#define MTB_PERIPHID5            MTB_PERIPHID_REG(MTB,1)
#define MTB_PERIPHID6            MTB_PERIPHID_REG(MTB,2)
#define MTB_PERIPHID7            MTB_PERIPHID_REG(MTB,3)
#define MTB_PERIPHID0             MTB_PERIPHID_REG(MTB,4)
#define MTB_PERIPHID1             MTB_PERIPHID_REG(MTB,5)
#define MTB_PERIPHID2             MTB_PERIPHID_REG(MTB,6)
#define MTB_PERIPHID3             MTB_PERIPHID_REG(MTB,7)
#define MTB_COMPID0               MTB_COMPID_REG(MTB,0)
#define MTB_COMPID1               MTB_COMPID_REG(MTB,1)
#define MTB_COMPID2               MTB_COMPID_REG(MTB,2)
#define MTB_COMPID3               MTB_COMPID_REG(MTB,3)
```

```
/* MTB - Register array accessors */

#define MTB_PERIPHID(index)          MTB_PERIPHID_REG(MTB,index)
#define MTB_COMPID(index)            MTB_COMPID_REG(MTB,index)

/*!
 * @}

 */ /* end of group MTB_Register_Accessor_Macros */
```

```
/*!
 * @}

 */ /* end of group MTB_Peripheral_Access_Layer */
```

```
/*
-----  
-- MTBDWT Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup MTBDWT_Peripheral_Access_Layer MTBDWT Peripheral Access Layer  
 * @{  
 */
```

```
/** MTBDWT - Register Layout Typedef */  
  
typedef struct {  
    __I uint32_t CTRL;           /**< MTB DWT Control Register, offset: 0x0 */  
    uint8_t RESERVED_0[28];  
    struct {                  /* offset: 0x20, array step: 0x10 */  
        /*  
         *  
         */  
    };  
};
```

```

    __IO uint32_t COMP;           /**< MTB_DWT Comparator Register, array offset: 0x20, array
step: 0x10 */

    __IO uint32_t MASK;          /**< MTB_DWT Comparator Mask Register, array offset: 0x24,
array step: 0x10 */

    __IO uint32_t FCT;           /**< MTB_DWT Comparator Function Register 0..MTB_DWT
Comparator Function Register 1, array offset: 0x28, array step: 0x10 */

    uint8_t RESERVED_0[4];

} COMPARATOR[2];

    uint8_t RESERVED_1[448];

__IO uint32_t TBCTRL;           /**< MTB_DWT Trace Buffer Control Register, offset: 0x200 */

    uint8_t RESERVED_2[3524];

__I uint32_t DEVICECFG;         /**< Device Configuration Register, offset: 0xFC8 */

__I uint32_t DEVICETYPID;       /**< Device Type Identifier Register, offset: 0xFCC */

__I uint32_t PERIPHID[8];       /**< Peripheral ID Register, array offset: 0xFD0, array step: 0x4
*/
    __I uint32_t COMPID[4];        /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
}

} MTBDWT_Type, *MTBDWT_MemMapPtr;

```

```

/*
-----*
-- MTBDWT - Register accessor macros
----- */

```

```

/*!
* @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros
* @{
*/

```

```

/* MTBDWT - Register accessors */

```

```
#define MTBDWT_CTRL_REG(base) ((base)->CTRL)
#define MTBDWT_COMP_REG(base,index) ((base)->COMPARATOR[index].COMP)
#define MTBDWT_COMP_COUNT 2
#define MTBDWT_MASK_REG(base,index) ((base)->COMPARATOR[index].MASK)
#define MTBDWT_MASK_COUNT 2
#define MTBDWT_FCT_REG(base,index) ((base)->COMPARATOR[index].FCT)
#define MTBDWT_FCT_COUNT 2
#define MTBDWT_TBCTRL_REG(base) ((base)->TBCTRL)
#define MTBDWT_DEVICECFG_REG(base) ((base)->DEVICECFG)
#define MTBDWT_DEVICETYPID_REG(base) ((base)->DEVICETYPID)
#define MTBDWT_PERIPHID_REG(base,index) ((base)->PERIPHID[index])
#define MTBDWT_PERIPHID_COUNT 8
#define MTBDWT_COMPID_REG(base,index) ((base)->COMPID[index])
#define MTBDWT_COMPID_COUNT 4
```

```
/*
 * @}
 */
/* /* end of group MTBDWT_Register_Accessor_Macros */
```

```
/*
-----  
-- MTBDWT Register Masks  
----- */
```

```
/*
 * @addtogroup MTBDWT_Register_Masks MTBDWT Register Masks
 * @{
 */
*/
```

```

/* CTRL Bit Fields */

#define MTBDWT_CTRL_DWTCFGCTRL_MASK      0xFFFFFFFFu
#define MTBDWT_CTRL_DWTCFGCTRL_SHIFT     0
#define MTBDWT_CTRL_DWTCFGCTRL_WIDTH     28
#define MTBDWT_CTRL_DWTCFGCTRL(x)        (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_DWTCFGCTRL_SHIFT))&MTBDWT_CTRL_DWTCFGCTRL_MASK)

#define MTBDWT_CTRL_NUMCMP_MASK          0xF0000000u
#define MTBDWT_CTRL_NUMCMP_SHIFT         28
#define MTBDWT_CTRL_NUMCMP_WIDTH         4
#define MTBDWT_CTRL_NUMCMP(x)           (((uint32_t)((uint32_t)(x))<<MTBDWT_CTRL_NUMCMP_SHIFT))&MTBDWT_CTRL_NUMCMP_MASK)

/* COMP Bit Fields */

#define MTBDWT_COMP_COMP_MASK           0xFFFFFFFFu
#define MTBDWT_COMP_COMP_SHIFT          0
#define MTBDWT_COMP_COMP_WIDTH          32
#define MTBDWT_COMP_COMP(x)             (((uint32_t)((uint32_t)(x))<<MTBDWT_COMP_COMP_SHIFT))&MTBDWT_COMP_COMP_MASK)

/* MASK Bit Fields */

#define MTBDWT_MASK_MASK_MASK          0x1Fu
#define MTBDWT_MASK_MASK_SHIFT         0
#define MTBDWT_MASK_MASK_WIDTH          5
#define MTBDWT_MASK_MASK(x)            (((uint32_t)((uint32_t)(x))<<MTBDWT_MASK_MASK_SHIFT))&MTBDWT_MASK_MASK_MASK)

/* FCT Bit Fields */

#define MTBDWT_FCT_FUNCTION_MASK       0xFu
#define MTBDWT_FCT_FUNCTION_SHIFT      0
#define MTBDWT_FCT_FUNCTION_WIDTH      4
#define MTBDWT_FCT_FUNCTION(x)         (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_FUNCTION_SHIFT))&MTBDWT_FCT_FUNCTION_MASK)

#define MTBDWT_FCT_DATAVMATCH_MASK     0x100u

```

```

#define MTBDWT_FCT_DATAVMATCH_SHIFT      8
#define MTBDWT_FCT_DATAVMATCH_WIDTH      1
#define MTBDWT_FCT_DATAVMATCH(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVMATCH_SHIFT))&MTBDWT_FCT_DATAVMATCH_MASK

#define MTBDWT_FCT_DATAVSIZE_MASK        0xC00u
#define MTBDWT_FCT_DATAVSIZE_SHIFT       10
#define MTBDWT_FCT_DATAVSIZE_WIDTH       2
#define MTBDWT_FCT_DATAVSIZE(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVSIZE_SHIFT))&MTBDWT_FCT_DATAVSIZE_MASK

#define MTBDWT_FCT_DATAVADDR0_MASK       0xF000u
#define MTBDWT_FCT_DATAVADDR0_SHIFT      12
#define MTBDWT_FCT_DATAVADDR0_WIDTH      4
#define MTBDWT_FCT_DATAVADDR0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVADDR0_SHIFT))&MTBDWT_FCT_DATAVADDR0_MASK

#define MTBDWT_FCT_MATCHED_MASK         0x1000000u
#define MTBDWT_FCT_MATCHED_SHIFT        24
#define MTBDWT_FCT_MATCHED_WIDTH        1
#define MTBDWT_FCT_MATCHED(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_MATCHED_SHIFT))&MTBDWT_FCT_MATCHED_MASK

/* TBCTRL Bit Fields */

#define MTBDWT_TBCTRL_ACMP0_MASK        0x1u
#define MTBDWT_TBCTRL_ACMP0_SHIFT       0
#define MTBDWT_TBCTRL_ACMP0_WIDTH       1
#define MTBDWT_TBCTRL_ACMP0(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACMP0_SHIFT))&MTBDWT_TBCTRL_ACMP0_MASK

#define MTBDWT_TBCTRL_ACMP1_MASK        0x2u
#define MTBDWT_TBCTRL_ACMP1_SHIFT       1
#define MTBDWT_TBCTRL_ACMP1_WIDTH       1
#define MTBDWT_TBCTRL_ACMP1(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACMP1_SHIFT))&MTBDWT_TBCTRL_ACMP1_MASK

```

```

#define MTBDWT_TBCTRL_NUMCOMP_MASK      0xF0000000u
#define MTBDWT_TBCTRL_NUMCOMP_SHIFT     28
#define MTBDWT_TBCTRL_NUMCOMP_WIDTH     4
#define MTBDWT_TBCTRL_NUMCOMP(x)        (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_NUMCOMP_SHIFT))&MTBDWT_TBCTRL_NUMCOMP_M
ASK)

/* DEVICECFG Bit Fields */

#define MTBDWT_DEVICECFG_DEVICECFG_MASK    0xFFFFFFFFu
#define MTBDWT_DEVICECFG_DEVICECFG_SHIFT   0
#define MTBDWT_DEVICECFG_DEVICECFG_WIDTH   32
#define MTBDWT_DEVICECFG_DEVICECFG(x)       (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICECFG_DEVICECFG_SHIFT))&MTBDWT_DEVICECFG_DEVICE
CFG_MASK)

/* DEVICETYPID Bit Fields */

#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK 0xFFFFFFFFu
#define MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT 0
#define MTBDWT_DEVICETYPID_DEVICETYPID_WIDTH 32
#define MTBDWT_DEVICETYPID_DEVICETYPID(x)   (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT))&MTBDWT_DEVICETYPID_DE
VICETYPID_MASK)

/* PERIPHID Bit Fields */

#define MTBDWT_PERIPHID_PERIPHID_MASK    0xFFFFFFFFu
#define MTBDWT_PERIPHID_PERIPHID_SHIFT   0
#define MTBDWT_PERIPHID_PERIPHID_WIDTH   32
#define MTBDWT_PERIPHID_PERIPHID(x)       (((uint32_t)((uint32_t)(x))<<MTBDWT_PERIPHID_PERIPHID_SHIFT))&MTBDWT_PERIPHID_PERIPHID_M
ASK)

/* COMPID Bit Fields */

#define MTBDWT_COMPID_COMPID_MASK        0xFFFFFFFFu
#define MTBDWT_COMPID_COMPID_SHIFT       0
#define MTBDWT_COMPID_COMPID_WIDTH      32

```

```

#define MTBDWT_COMPID_COMPID(x)
(((uint32_t)((uint32_t)(x))<<MTBDWT_COMPID_COMPID_SHIFT))&MTBDWT_COMPID_COMPID_MASK

/*
 * @}
 */

/* /* end of group MTBDWT_Register_Masks */

/* MTBDWT - Peripheral instance base addresses */
/** Peripheral MTBDWT base address */
#define MTBDWT_BASE           (0xF0001000u)
/** Peripheral MTBDWT base pointer */
#define MTBDWT             ((MTBDWT_Type *)MTBDWT_BASE)
#define MTBDWT_BASE_PTR      (MTBDWT)
/** Array initializer of MTBDWT peripheral base addresses */
#define MTBDWT_BASE_ADDRS    { MTBDWT_BASE }
/** Array initializer of MTBDWT peripheral base pointers */
#define MTBDWT_BASE_PTRS     { MTBDWT }

/*
-----  

-- MTBDWT - Register accessor macros  

----- */
/*!  

* @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor macros  

* @{
*/

```

```

/* MTBDWT - Register instance definitions */

/* MTBDWT */

#define MTBDWT_CTRL           MTBDWT_CTRL_REG(MTBDWT)
#define MTBDWT_COMP0          MTBDWT_COMP_REG(MTBDWT,0)
#define MTBDWT_MASK0          MTBDWT_MASK_REG(MTBDWT,0)
#define MTBDWT_FCT0           MTBDWT_FCT_REG(MTBDWT,0)
#define MTBDWT_COMP1          MTBDWT_COMP_REG(MTBDWT,1)
#define MTBDWT_MASK1          MTBDWT_MASK_REG(MTBDWT,1)
#define MTBDWT_FCT1           MTBDWT_FCT_REG(MTBDWT,1)
#define MTBDWT_TBCTRL         MTBDWT_TBCTRL_REG(MTBDWT)
#define MTBDWT_DEVICECFG      MTBDWT_DEVICECFG_REG(MTBDWT)
#define MTBDWT_DEVICETYPID    MTBDWT_DEVICETYPID_REG(MTBDWT)
#define MTBDWT_PERIPHID4     MTBDWT_PERIPHID_REG(MTBDWT,0)
#define MTBDWT_PERIPHID5     MTBDWT_PERIPHID_REG(MTBDWT,1)
#define MTBDWT_PERIPHID6     MTBDWT_PERIPHID_REG(MTBDWT,2)
#define MTBDWT_PERIPHID7     MTBDWT_PERIPHID_REG(MTBDWT,3)
#define MTBDWT_PERIPHID0     MTBDWT_PERIPHID_REG(MTBDWT,4)
#define MTBDWT_PERIPHID1     MTBDWT_PERIPHID_REG(MTBDWT,5)
#define MTBDWT_PERIPHID2     MTBDWT_PERIPHID_REG(MTBDWT,6)
#define MTBDWT_PERIPHID3     MTBDWT_PERIPHID_REG(MTBDWT,7)
#define MTBDWT_COMPID0        MTBDWT_COMPID_REG(MTBDWT,0)
#define MTBDWT_COMPID1        MTBDWT_COMPID_REG(MTBDWT,1)
#define MTBDWT_COMPID2        MTBDWT_COMPID_REG(MTBDWT,2)
#define MTBDWT_COMPID3        MTBDWT_COMPID_REG(MTBDWT,3)

/* MTBDWT - Register array accessors */

#define MTBDWT_COMP(index)    MTBDWT_COMP_REG(MTBDWT,index)
#define MTBDWT_MASK(index)    MTBDWT_MASK_REG(MTBDWT,index)
#define MTBDWT_FCT(index)    MTBDWT_FCT_REG(MTBDWT,index)

```

```
#define MTBDWT_PERIPHID(index)          MTBDWT_PERIPHID_REG(MTBDWT,index)
#define MTBDWT_COMPID(index)              MTBDWT_COMPID_REG(MTBDWT,index)
```

```
/*!
 * @}
 */
/* end of group MTBDWT_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group MTBDWT_Peripheral_Access_Layer */
```

```
/*
-----  
-- NV Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup NV_Peripheral_Access_Layer NV Peripheral Access Layer  
 * @{  
 */
```

```
/** NV - Register Layout Typedef */
```

```
typedef struct {  
    __I uint8_t BACKKEY3;           /**< Backdoor Comparison Key 3., offset: 0x0 */  
    __I uint8_t BACKKEY2;           /**< Backdoor Comparison Key 2., offset: 0x1 */  
    __I uint8_t BACKKEY1;           /**< Backdoor Comparison Key 1., offset: 0x2 */  
    __I uint8_t BACKKEY0;           /**< Backdoor Comparison Key 0., offset: 0x3 */  
    __I uint8_t BACKKEY7;           /**< Backdoor Comparison Key 7., offset: 0x4 */
```

```

__I uint8_t BACKKEY6;           /**< Backdoor Comparison Key 6., offset: 0x5 */
__I uint8_t BACKKEY5;           /**< Backdoor Comparison Key 5., offset: 0x6 */
__I uint8_t BACKKEY4;           /**< Backdoor Comparison Key 4., offset: 0x7 */
__I uint8_t FPROT3;            /**< Non-volatile P-Flash Protection 1 - Low Register, offset: 0x8
*/
__I uint8_t FPROT2;            /**< Non-volatile P-Flash Protection 1 - High Register, offset: 0x9
*/
__I uint8_t FPROT1;            /**< Non-volatile P-Flash Protection 0 - Low Register, offset: 0xA
*/
__I uint8_t FPROT0;            /**< Non-volatile P-Flash Protection 0 - High Register, offset: 0xB
*/
__I uint8_t FSEC;              /**< Non-volatile Flash Security Register, offset: 0xC */
__I uint8_t FOPT;              /**< Non-volatile Flash Option Register, offset: 0xD */

} NV_Type, *NV_MemMapPtr;

```

```

/*
-----*
-- NV - Register accessor macros
----- */

```

```
/*!
```

```
* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros
* @{
* /

```

```
/* NV - Register accessors */
```

```
#define NV_BACKKEY3_REG(base)      ((base)->BACKKEY3)
#define NV_BACKKEY2_REG(base)      ((base)->BACKKEY2)
#define NV_BACKKEY1_REG(base)      ((base)->BACKKEY1)
#define NV_BACKKEY0_REG(base)      ((base)->BACKKEY0)
```

```
#define NV_BACKKEY7_REG(base)          ((base)->BACKKEY7)
#define NV_BACKKEY6_REG(base)          ((base)->BACKKEY6)
#define NV_BACKKEY5_REG(base)          ((base)->BACKKEY5)
#define NV_BACKKEY4_REG(base)          ((base)->BACKKEY4)
#define NV_FPROT3_REG(base)           ((base)->FPROT3)
#define NV_FPROT2_REG(base)           ((base)->FPROT2)
#define NV_FPROT1_REG(base)           ((base)->FPROT1)
#define NV_FPROT0_REG(base)           ((base)->FPROT0)
#define NV_FSEC_REG(base)             ((base)->FSEC)
#define NV_FOPT_REG(base)             ((base)->FOPT)
```

```
/*
 * @}
 */
/* /* end of group NV_Register_Accessor_Macros */
```

```
/*
-----  
-- NV Register Masks  
----- */
```

```
/*
 * @addtogroup NV_Register_Masks NV Register Masks
 * @{
 */

```

```
/* BACKKEY3 Bit Fields */

#define NV_BACKKEY3_KEY_MASK          0xFFu
#define NV_BACKKEY3_KEY_SHIFT         0
#define NV_BACKKEY3_KEY_WIDTH         8
```

```

#define NV_BACKKEY3_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY3_KEY_SHIFT))&NV_BACKKEY3_KEY_MASK)

/* BACKKEY2 Bit Fields */

#define NV_BACKKEY2_KEY_MASK          0xFFu
#define NV_BACKKEY2_KEY_SHIFT         0
#define NV_BACKKEY2_KEY_WIDTH         8

#define NV_BACKKEY2_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY2_KEY_SHIFT))&NV_BACKKEY2_KEY_MASK)

/* BACKKEY1 Bit Fields */

#define NV_BACKKEY1_KEY_MASK          0xFFu
#define NV_BACKKEY1_KEY_SHIFT         0
#define NV_BACKKEY1_KEY_WIDTH         8

#define NV_BACKKEY1_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY1_KEY_SHIFT))&NV_BACKKEY1_KEY_MASK)

/* BACKKEY0 Bit Fields */

#define NV_BACKKEY0_KEY_MASK          0xFFu
#define NV_BACKKEY0_KEY_SHIFT         0
#define NV_BACKKEY0_KEY_WIDTH         8

#define NV_BACKKEY0_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY0_KEY_SHIFT))&NV_BACKKEY0_KEY_MASK)

/* BACKKEY7 Bit Fields */

#define NV_BACKKEY7_KEY_MASK          0xFFu
#define NV_BACKKEY7_KEY_SHIFT         0
#define NV_BACKKEY7_KEY_WIDTH         8

#define NV_BACKKEY7_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY7_KEY_SHIFT))&NV_BACKKEY7_KEY_MASK)

/* BACKKEY6 Bit Fields */

#define NV_BACKKEY6_KEY_MASK          0xFFu
#define NV_BACKKEY6_KEY_SHIFT         0
#define NV_BACKKEY6_KEY_WIDTH         8

```

```

#define NV_BACKKEY6_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY6_KEY_SHIFT))&NV_BACKKEY6_KEY_MASK)

/* BACKKEY5 Bit Fields */

#define NV_BACKKEY5_KEY_MASK          0xFFu
#define NV_BACKKEY5_KEY_SHIFT         0
#define NV_BACKKEY5_KEY_WIDTH         8

#define NV_BACKKEY5_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY5_KEY_SHIFT))&NV_BACKKEY5_KEY_MASK)

/* BACKKEY4 Bit Fields */

#define NV_BACKKEY4_KEY_MASK          0xFFu
#define NV_BACKKEY4_KEY_SHIFT         0
#define NV_BACKKEY4_KEY_WIDTH         8

#define NV_BACKKEY4_KEY(x)
(((uint8_t)((uint8_t)(x)<<NV_BACKKEY4_KEY_SHIFT))&NV_BACKKEY4_KEY_MASK)

/* FPROT3 Bit Fields */

#define NV_FPROT3_PROT_MASK          0xFFu
#define NV_FPROT3_PROT_SHIFT         0
#define NV_FPROT3_PROT_WIDTH         8

#define NV_FPROT3_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT3_PROT_SHIFT))&NV_FPROT3_PROT_MASK)

/* FPROT2 Bit Fields */

#define NV_FPROT2_PROT_MASK          0xFFu
#define NV_FPROT2_PROT_SHIFT         0
#define NV_FPROT2_PROT_WIDTH         8

#define NV_FPROT2_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT2_PROT_SHIFT))&NV_FPROT2_PROT_MASK)

/* FPROT1 Bit Fields */

#define NV_FPROT1_PROT_MASK          0xFFu
#define NV_FPROT1_PROT_SHIFT         0
#define NV_FPROT1_PROT_WIDTH         8

```

```

#define NV_FPROT1_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT1_PROT_SHIFT))&NV_FPROT1_PROT_MASK)

/* FPROT0 Bit Fields */

#define NV_FPROT0_PROT_MASK          0xFFu
#define NV_FPROT0_PROT_SHIFT         0
#define NV_FPROT0_PROT_WIDTH         8

#define NV_FPROT0_PROT(x)
(((uint8_t)((uint8_t)(x)<<NV_FPROT0_PROT_SHIFT))&NV_FPROT0_PROT_MASK)

/* FSEC Bit Fields */

#define NV_FSEC_SEC_MASK            0x3u
#define NV_FSEC_SEC_SHIFT           0
#define NV_FSEC_SEC_WIDTH           2

#define NV_FSEC_SEC(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_SEC_SHIFT))&NV_FSEC_SEC_MASK)

#define NV_FSEC_FSLACC_MASK          0xCu
#define NV_FSEC_FSLACC_SHIFT         2
#define NV_FSEC_FSLACC_WIDTH         2

#define NV_FSEC_FSLACC(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_FSLACC_SHIFT))&NV_FSEC_FSLACC_MASK)

#define NV_FSEC_MEEN_MASK            0x30u
#define NV_FSEC_MEEN_SHIFT           4
#define NV_FSEC_MEEN_WIDTH           2

#define NV_FSEC_MEEN(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_MEEN_SHIFT))&NV_FSEC_MEEN_MASK)

#define NV_FSEC_KEYEN_MASK           0xC0u
#define NV_FSEC_KEYEN_SHIFT          6
#define NV_FSEC_KEYEN_WIDTH           2

#define NV_FSEC_KEYEN(x)
(((uint8_t)((uint8_t)(x)<<NV_FSEC_KEYEN_SHIFT))&NV_FSEC_KEYEN_MASK)

/* FOPT Bit Fields */

#define NV_FOPT_LPBOOT0_MASK          0x1u

```

```

#define NV_FOPT_LPBOOT0_SHIFT          0
#define NV_FOPT_LPBOOT0_WIDTH          1
#define NV_FOPT_LPBOOT0(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT0_SHIFT))&NV_FOPT_LPBOOT0_MASK)

#define NV_FOPT_NMI_DIS_MASK          0x4u
#define NV_FOPT_NMI_DIS_SHIFT         2
#define NV_FOPT_NMI_DIS_WIDTH         1
#define NV_FOPT_NMI_DIS(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_NMI_DIS_SHIFT))&NV_FOPT_NMI_DIS_MASK)

#define NV_FOPT_RESET_PIN_CFG_MASK     0x8u
#define NV_FOPT_RESET_PIN_CFG_SHIFT   3
#define NV_FOPT_RESET_PIN_CFG_WIDTH   1
#define NV_FOPT_RESET_PIN_CFG(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_RESET_PIN_CFG_SHIFT))&NV_FOPT_RESET_PIN_CFG_MASK)

#define NV_FOPT_LPBOOT1_MASK          0x10u
#define NV_FOPT_LPBOOT1_SHIFT         4
#define NV_FOPT_LPBOOT1_WIDTH          1
#define NV_FOPT_LPBOOT1(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT1_SHIFT))&NV_FOPT_LPBOOT1_MASK)

#define NV_FOPT_FAST_INIT_MASK         0x20u
#define NV_FOPT_FAST_INIT_SHIFT        5
#define NV_FOPT_FAST_INIT_WIDTH        1
#define NV_FOPT_FAST_INIT(x)
(((uint8_t)((uint8_t)(x)<<NV_FOPT_FAST_INIT_SHIFT))&NV_FOPT_FAST_INIT_MASK)

/*
 * @}
 */ /* end of group NV_Register_Masks */

```

```

/* NV - Peripheral instance base addresses */

/** Peripheral FTFA_FlashConfig base address */

#define FTFA_FlashConfig_BASE          (0x400u)

/** Peripheral FTFA_FlashConfig base pointer */

#define FTFA_FlashConfig             ((NV_Type *)FTFA_FlashConfig_BASE)

#define FTFA_FlashConfig_BASE_PTR    (FTFA_FlashConfig)

/** Array initializer of NV peripheral base addresses */

#define NV_BASE_ADDRS               { FTFA_FlashConfig_BASE }

/** Array initializer of NV peripheral base pointers */

#define NV_BASE_PTRS                { FTFA_FlashConfig }

```

```

/*
-----  

-- NV - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros  

* @{
* /  


```

```

/* NV - Register instance definitions */

/* FTFA_FlashConfig */

#define NV_BACKKEY3           NV_BACKKEY3_REG(FTFA_FlashConfig)  

#define NV_BACKKEY2           NV_BACKKEY2_REG(FTFA_FlashConfig)  

#define NV_BACKKEY1           NV_BACKKEY1_REG(FTFA_FlashConfig)  

#define NV_BACKKEY0           NV_BACKKEY0_REG(FTFA_FlashConfig)  

#define NV_BACKKEY7           NV_BACKKEY7_REG(FTFA_FlashConfig)  

#define NV_BACKKEY6           NV_BACKKEY6_REG(FTFA_FlashConfig)

```

```
#define NV_BACKKEY5           NV_BACKKEY5_REG(FTFA_FlashConfig)
#define NV_BACKKEY4           NV_BACKKEY4_REG(FTFA_FlashConfig)
#define NV_FPROT3             NV_FPROT3_REG(FTFA_FlashConfig)
#define NV_FPROT2             NV_FPROT2_REG(FTFA_FlashConfig)
#define NV_FPROT1             NV_FPROT1_REG(FTFA_FlashConfig)
#define NV_FPROT0             NV_FPROT0_REG(FTFA_FlashConfig)
#define NV_FSEC               NV_FSEC_REG(FTFA_FlashConfig)
#define NV_FOPT               NV_FOPT_REG(FTFA_FlashConfig)
```

```
/*!
 * @}
 */
/* /* end of group NV_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* /* end of group NV_Peripheral_Access_Layer */
```

```
/*
-----  
-- OSC Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup OSC_Peripheral_Access_Layer OSC Peripheral Access Layer  
 * @{  
 */
```

```
/** OSC - Register Layout Typedef */
```

```
typedef struct {  
    __IO uint8_t CR;           /*< OSC Control Register, offset: 0x0 */  
} OSC_Type, *OSC_MemMapPtr;
```

```
/* -----  
-- OSC - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros  
* @{  
*/
```

```
/* OSC - Register accessors */  
#define OSC_CR_REG(base) ((base)->CR)
```

```
/*!  
* @}  
* ///* end of group OSC_Register_Accessor_Macros */
```

```
/* -----  
-- OSC Register Masks  
----- */
```

```
/*!  
* @addtogroup OSC_Register_Masks OSC Register Masks  
* @{
```

*/

```
/* CR Bit Fields */

#define OSC_CR_SC16P_MASK          0x1u
#define OSC_CR_SC16P_SHIFT         0
#define OSC_CR_SC16P_WIDTH          1
#define OSC_CR_SC16P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC16P_SHIFT))&OSC_CR_SC16P_MASK)

#define OSC_CR_SC8P_MASK           0x2u
#define OSC_CR_SC8P_SHIFT          1
#define OSC_CR_SC8P_WIDTH          1
#define OSC_CR_SC8P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC8P_SHIFT))&OSC_CR_SC8P_MASK)

#define OSC_CR_SC4P_MASK           0x4u
#define OSC_CR_SC4P_SHIFT          2
#define OSC_CR_SC4P_WIDTH          1
#define OSC_CR_SC4P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC4P_SHIFT))&OSC_CR_SC4P_MASK)

#define OSC_CR_SC2P_MASK           0x8u
#define OSC_CR_SC2P_SHIFT          3
#define OSC_CR_SC2P_WIDTH          1
#define OSC_CR_SC2P(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_SC2P_SHIFT))&OSC_CR_SC2P_MASK)

#define OSC_CR_EREFSTEN_MASK        0x20u
#define OSC_CR_EREFSTEN_SHIFT       5
#define OSC_CR_EREFSTEN_WIDTH       1
#define OSC_CR_EREFSTEN(x) (((uint8_t)((uint8_t)(x))<<OSC_CR_EREFSTEN_SHIFT))&OSC_CR_EREFSTEN_MASK)

#define OSC_CR_ERCLKEN_MASK         0x80u
#define OSC_CR_ERCLKEN_SHIFT        7
```

```
#define OSC_CR_ERCLKEN_WIDTH          1
#define OSC_CR_ERCLKEN(x)
((uint8_t)((uint8_t)(x)<<OSC_CR_ERCLKEN_SHIFT))&OSC_CR_ERCLKEN_MASK)
```

```
/*!
 * @}
 */
/* /* end of group OSC_Register_Masks */
```

```
/* OSC - Peripheral instance base addresses */
/** Peripheral OSC0 base address */
#define OSC0_BASE          (0x40065000u)
/** Peripheral OSC0 base pointer */
#define OSC0           ((OSC_Type *)OSCO_BASE)
#define OSC0_BASE_PTR      (OSCO)
/** Array initializer of OSC peripheral base addresses */
#define OSC_BASE_ADDRS     { OSC0_BASE }
/** Array initializer of OSC peripheral base pointers */
#define OSC_BASE_PTRS       { OSC0 }
```

```
/*
-----  
-- OSC - Register accessor macros  
----- */
```

```
/*!
 * @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor macros
 * @{
 */

```

```

/* OSC - Register instance definitions */

/* OSC0 */

#define OSC0_CR           OSC_CR_REG(OSC0)

/*
 * @}
 */

/* /* end of group OSC_Register_Accessor_Macros */

/*
 * @}
 */

/* /* end of group OSC_Peripheral_Access_Layer */

/*
-----  

-- PIT Peripheral Access Layer  

----- */

```

```

/*
 * @addtogroup PIT_Peripheral_Access_Layer PIT Peripheral Access Layer
 * @{
 */

/** PIT - Register Layout Typedef */

typedef struct {

    __IO uint32_t MCR;           /**< PIT Module Control Register, offset: 0x0 */

    uint8_t RESERVED_0[220];

    __I uint32_t LTMR64H;        /**< PIT Upper Lifetime Timer Register, offset: 0xE0 */
}

```

```

__I uint32_t LTMR64L;           /**< PIT Lower Lifetime Timer Register, offset: 0xE4 */

uint8_t RESERVED_1[24];

struct {                         /* offset: 0x100, array step: 0x10 */

    __IO uint32_t LDVAL;          /**< Timer Load Value Register, array offset: 0x100, array step:
0x10 */

    __I uint32_t CVAL;           /**< Current Timer Value Register, array offset: 0x104, array step:
0x10 */

    __IO uint32_t TCTRL;          /**< Timer Control Register, array offset: 0x108, array step: 0x10

    __IO uint32_t TFLG;           /**< Timer Flag Register, array offset: 0x10C, array step: 0x10

} CHANNEL[2];

} PIT_Type, *PIT_MemMapPtr;

```

```

/*
-----  

-- PIT - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros  

* @{
* /  


```

```

/* PIT - Register accessors */  

#define PIT_MCR_REG(base)          ((base)->MCR)  

#define PIT_LTMR64H_REG(base)       ((base)->LTMR64H)  

#define PIT_LTMR64L_REG(base)       ((base)->LTMR64L)  

#define PIT_LDVAL_REG(base,index)   ((base)->CHANNEL[index].LDVAL)  

#define PIT_LDVAL_COUNT            2  

#define PIT_CVAL_REG(base,index)    ((base)->CHANNEL[index].CVAL)

```

```
#define PIT_CVAL_COUNT          2
#define PIT_TCTRL_REG(base,index) ((base)->CHANNEL[index].TCTRL)
#define PIT_TCTRL_COUNT          2
#define PIT_TFLG_REG(base,index)  ((base)->CHANNEL[index].TFLG)
#define PIT_TFLG_COUNT          2

/*!
 * @}
 */
/* /* end of group PIT_Register_Accessor_Macros */
```

```
/*
-----  
-- PIT Register Masks  
----- */
```

```
/*!  
 * @addtogroup PIT_Register_Masks PIT Register Masks  
 * @{  
 */
```

```
/* MCR Bit Fields */

#define PIT_MCR_FRZ_MASK          0x1u
#define PIT_MCR_FRZ_SHIFT         0
#define PIT_MCR_FRZ_WIDTH         1
#define PIT_MCR_FRZ(x)            (((uint32_t)((uint32_t)(x)<<PIT_MCR_FRZ_SHIFT))&PIT_MCR_FRZ_MASK)
#define PIT_MCR_MDIS_MASK          0x2u
#define PIT_MCR_MDIS_SHIFT         1
#define PIT_MCR_MDIS_WIDTH         1
```

```

#define PIT_MCR_MDIS(x)
(((uint32_t) (((uint32_t)(x))<<PIT_MCR_MDIS_SHIFT))&PIT_MCR_MDIS_MASK)

/* LTMR64H Bit Fields */

#define PIT_LTMR64H_LTH_MASK          0xFFFFFFFFu
#define PIT_LTMR64H_LTH_SHIFT         0
#define PIT_LTMR64H_LTH_WIDTH         32
#define PIT_LTMR64H_LTH(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LTMR64H_LTH_SHIFT))&PIT_LTMR64H_LTH_MASK)

/* LTMR64L Bit Fields */

#define PIT_LTMR64L_LTL_MASK          0xFFFFFFFFu
#define PIT_LTMR64L_LTL_SHIFT         0
#define PIT_LTMR64L_LTL_WIDTH         32
#define PIT_LTMR64L_LTL(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LTMR64L_LTL_SHIFT))&PIT_LTMR64L_LTL_MASK)

/* LDVAL Bit Fields */

#define PIT_LDVAL_TSV_MASK           0xFFFFFFFFu
#define PIT_LDVAL_TSV_SHIFT          0
#define PIT_LDVAL_TSV_WIDTH          32
#define PIT_LDVAL_TSV(x)
(((uint32_t) (((uint32_t)(x))<<PIT_LDVAL_TSV_SHIFT))&PIT_LDVAL_TSV_MASK)

/* CVAL Bit Fields */

#define PIT_CVAL_TVL_MASK            0xFFFFFFFFu
#define PIT_CVAL_TVL_SHIFT           0
#define PIT_CVAL_TVL_WIDTH           32
#define PIT_CVAL_TVL(x)
(((uint32_t) (((uint32_t)(x))<<PIT_CVAL_TVL_SHIFT))&PIT_CVAL_TVL_MASK)

/* TCTRL Bit Fields */

#define PIT_TCTRL_TEN_MASK           0x1u
#define PIT_TCTRL_TEN_SHIFT          0
#define PIT_TCTRL_TEN_WIDTH          1

```

```

#define PIT_TCTRL_TEN(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_TEN_SHIFT))&PIT_TCTRL_TEN_MASK)

#define PIT_TCTRL_TIE_MASK          0x2u
#define PIT_TCTRL_TIE_SHIFT         1
#define PIT_TCTRL_TIE_WIDTH         1

#define PIT_TCTRL_TIE(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_TIE_SHIFT))&PIT_TCTRL_TIE_MASK)

#define PIT_TCTRL_CHN_MASK          0x4u
#define PIT_TCTRL_CHN_SHIFT         2
#define PIT_TCTRL_CHN_WIDTH         1

#define PIT_TCTRL_CHN(x)
(((uint32_t)((uint32_t)(x)<<PIT_TCTRL_CHN_SHIFT))&PIT_TCTRL_CHN_MASK)

/* TFLG Bit Fields */

#define PIT_TFLG_TIF_MASK          0x1u
#define PIT_TFLG_TIF_SHIFT          0
#define PIT_TFLG_TIF_WIDTH          1

#define PIT_TFLG_TIF(x)
(((uint32_t)((uint32_t)(x)<<PIT_TFLG_TIF_SHIFT))&PIT_TFLG_TIF_MASK)

/*
 */
*! 
* @}

/* /* end of group PIT_Register_Masks */

```

```

/* PIT - Peripheral instance base addresses */

/** Peripheral PIT base address */

#define PIT_BASE                  (0x40037000u)

/** Peripheral PIT base pointer */

#define PIT                      ((PIT_Type *)PIT_BASE)

#define PIT_BASE_PTR             (PIT)

```

```

/** Array initializer of PIT peripheral base addresses */
#define PIT_BASE_ADDRS          { PIT_BASE }

/** Array initializer of PIT peripheral base pointers */
#define PIT_BASE_PTRS           { PIT }

/*
-----  

-- PIT - Register accessor macros  

----- */
/*!  

* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros  

* @{
* /  

  

/* PIT - Register instance definitions */  

/* PIT */  

#define PIT_MCR                  PIT_MCR_REG(PIT)  

#define PIT_LTMR64H               PIT_LTMR64H_REG(PIT)  

#define PIT_LTMR64L               PIT_LTMR64L_REG(PIT)  

#define PIT_LDVAL0                PIT_LDVAL_REG(PIT,0)  

#define PIT_CVAL0                 PIT_CVAL_REG(PIT,0)  

#define PIT_TCTRL0                PIT_TCTRL_REG(PIT,0)  

#define PIT_TFLG0                 PIT_TFLG_REG(PIT,0)  

#define PIT_LDVAL1                PIT_LDVAL_REG(PIT,1)  

#define PIT_CVAL1                 PIT_CVAL_REG(PIT,1)  

#define PIT_TCTRL1                PIT_TCTRL_REG(PIT,1)  

#define PIT_TFLG1                 PIT_TFLG_REG(PIT,1)

```

```
/* PIT - Register array accessors */

#define PIT_LDVAL(index)          PIT_LDVAL_REG(PIT,index)
#define PIT_CVAL(index)           PIT_CVAL_REG(PIT,index)
#define PIT_TCTRL(index)          PIT_TCTRL_REG(PIT,index)
#define PIT_TFLG(index)           PIT_TFLG_REG(PIT,index)
```

```
/*
 * @}
 */
/* end of group PIT_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* end of group PIT_Peripheral_Access_Layer */
```

```
/*
-----  
-- PMC Peripheral Access Layer  
----- */
```

```
/*
 * @addtogroup PMC_Peripheral_Access_Layer PMC Peripheral Access Layer
 * @{
 */

```

```
/** PMC - Register Layout Typedef */

typedef struct {
    _IO uint8_t LVDSC1;           /**< Low Voltage Detect Status And Control 1 register, offset: 0x0
*/
```

```
    __IO uint8_t LVDSC2;           /**< Low Voltage Detect Status And Control 2 register, offset: 0x1
*/
    __IO uint8_t REGSC;           /**< Regulator Status And Control register, offset: 0x2 */
} PMC_Type, *PMC_MemMapPtr;
```

```
/*
-----*
-- PMC - Register accessor macros
----- */
```

```
/*!
* @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros
* @{
*/

```

```
/* PMC - Register accessors */
#define PMC_LVDSC1_REG(base)      ((base)->LVDSC1)
#define PMC_LVDSC2_REG(base)      ((base)->LVDSC2)
#define PMC_REGSC_REG(base)       ((base)->REGSC)
```

```
/*!
* @}
*/
/* /* end of group PMC_Register_Accessor_Macros */
```

```
/*
-----*
-- PMC Register Masks
----- */
```

```

/*!
 * @addtogroup PMC_Register_Masks PMC Register Masks
 * @{
 */

/* LVDSC1 Bit Fields */

#define PMC_LVDSC1_LVDV_MASK          0x3u
#define PMC_LVDSC1_LVDV_SHIFT         0
#define PMC_LVDSC1_LVDV_WIDTH         2
#define PMC_LVDSC1_LVDV(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDV_SHIFT))&PMC_LVDSC1_LVDV_MASK)

#define PMC_LVDSC1_LVDRE_MASK         0x10u
#define PMC_LVDSC1_LVDRE_SHIFT        4
#define PMC_LVDSC1_LVDRE_WIDTH        1
#define PMC_LVDSC1_LVDRE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDRE_SHIFT))&PMC_LVDSC1_LVDRE_MASK)

#define PMC_LVDSC1_LVDIE_MASK         0x20u
#define PMC_LVDSC1_LVDIE_SHIFT        5
#define PMC_LVDSC1_LVDIE_WIDTH        1
#define PMC_LVDSC1_LVDIE(x)           (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDIE_SHIFT))&PMC_LVDSC1_LVDIE_MASK)

#define PMC_LVDSC1_LVDACK_MASK        0x40u
#define PMC_LVDSC1_LVDACK_SHIFT       6
#define PMC_LVDSC1_LVDACK_WIDTH       1
#define PMC_LVDSC1_LVDACK(x)          (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDACK_SHIFT))&PMC_LVDSC1_LVDACK_MASK)

#define PMC_LVDSC1_LVDF_MASK          0x80u
#define PMC_LVDSC1_LVDF_SHIFT         7
#define PMC_LVDSC1_LVDF_WIDTH         1

```

```

#define PMC_LVDSC1_LVDF(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC1_LVDF_SHIFT))&PMC_LVDSC1_LVDF_MASK)

/* LVDSC2 Bit Fields */

#define PMC_LVDSC2_LVWV_MASK          0x3u
#define PMC_LVDSC2_LVWV_SHIFT         0
#define PMC_LVDSC2_LVWV_WIDTH         2
#define PMC_LVDSC2_LVWV(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWV_SHIFT))&PMC_LVDSC2_LVWV_MASK)

#define PMC_LVDSC2_LVWIE_MASK         0x20u
#define PMC_LVDSC2_LVWIE_SHIFT        5
#define PMC_LVDSC2_LVWIE_WIDTH        1
#define PMC_LVDSC2_LVWIE(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWIE_SHIFT))&PMC_LVDSC2_LVWIE_MASK)

#define PMC_LVDSC2_LVWACK_MASK        0x40u
#define PMC_LVDSC2_LVWACK_SHIFT       6
#define PMC_LVDSC2_LVWACK_WIDTH       1
#define PMC_LVDSC2_LVWACK(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWACK_SHIFT))&PMC_LVDSC2_LVWACK_MASK)

#define PMC_LVDSC2_LVWF_MASK          0x80u
#define PMC_LVDSC2_LVWF_SHIFT         7
#define PMC_LVDSC2_LVWF_WIDTH         1
#define PMC_LVDSC2_LVWF(x)
(((uint8_t)((uint8_t)(x)<<PMC_LVDSC2_LVWF_SHIFT))&PMC_LVDSC2_LVWF_MASK)

/* REGSC Bit Fields */

#define PMC_REGSC_BGBE_MASK           0x1u
#define PMC_REGSC_BGBE_SHIFT          0
#define PMC_REGSC_BGBE_WIDTH          1
#define PMC_REGSC_BGBE(x)
(((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGBE_SHIFT))&PMC_REGSC_BGBE_MASK)

#define PMC_REGSC_REGONS_MASK         0x4u
#define PMC_REGSC_REGONS_SHIFT        2

```

```

#define PMC_REGSC_REGONS_WIDTH          1
#define PMC_REGSC_REGONS(x)             (((uint8_t)((uint8_t)(x)<<PMC_REGSC_REGONS_SHIFT))&PMC_REGSC_REGONS_MASK)
#define PMC_REGSC_ACKISO_MASK           0x8u
#define PMC_REGSC_ACKISO_SHIFT          3
#define PMC_REGSC_ACKISO_WIDTH          1
#define PMC_REGSC_ACKISO(x)             (((uint8_t)((uint8_t)(x)<<PMC_REGSC_ACKISO_SHIFT))&PMC_REGSC_ACKISO_MASK)
#define PMC_REGSC_BGEN_MASK             0x10u
#define PMC_REGSC_BGEN_SHIFT            4
#define PMC_REGSC_BGEN_WIDTH            1
#define PMC_REGSC_BGEN(x)               (((uint8_t)((uint8_t)(x)<<PMC_REGSC_BGEN_SHIFT))&PMC_REGSC_BGEN_MASK)

```

```

/*
 * @}
 */
/* end of group PMC_Register_Masks */

```

```

/* PMC - Peripheral instance base addresses */
/** Peripheral PMC base address */
#define PMC_BASE                      (0x4007D000u)
/** Peripheral PMC base pointer */
#define PMC                         ((PMC_Type *)PMC_BASE)
#define PMC_BASE_PTR                  (PMC)
/** Array initializer of PMC peripheral base addresses */
#define PMC_BASE_ADDRS                { PMC_BASE }
/** Array initializer of PMC peripheral base pointers */
#define PMC_BASE_PTRS                 { PMC }

```

```
/*
-- PMC - Register accessor macros
----- */
/*!
* @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor macros
* @{
*/
/* PMC - Register instance definitions */
/* PMC */
#define PMC_LVDSC1          PMC_LVDSC1_REG(PMC)
#define PMC_LVDSC2          PMC_LVDSC2_REG(PMC)
#define PMC_REGSC           PMC_REGSC_REG(PMC)

/*!
* @}
*/
/* end of group PMC_Register_Accessor_Macros */

/*!
* @}
*/
/* end of group PMC_Peripheral_Access_Layer */

/*
-- PORT Peripheral Access Layer
----- */

```

```

/*!

* @addtogroup PORT_Peripheral_Access_Layer PORT Peripheral Access Layer

* @{

*/



/** PORT - Register Layout Typedef */

typedef struct {

    __IO uint32_t PCR[32];           /**< Pin Control Register n, array offset: 0x0, array step: 0x4 */

    __O uint32_t GPCLR;             /**< Global Pin Control Low Register, offset: 0x80 */

    __O uint32_t GPCHR;             /**< Global Pin Control High Register, offset: 0x84 */

    uint8_t RESERVED_0[24];

    __IO uint32_t ISFR;             /**< Interrupt Status Flag Register, offset: 0xA0 */

} PORT_Type, *PORT_MemMapPtr;

```

```

/*
-----  

-- PORT - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros  

* @{
*/
```

```

/* PORT - Register accessors */

#define PORT_PCR_REG(base,index)      ((base)->PCR[index])  

#define PORT_PCR_COUNT                32  

#define PORT_GPCLR_REG(base)          ((base)->GPCLR)
```

```
#define PORT_GPCHR_REG(base)          ((base)->GPCHR)
#define PORT_ISFR_REG(base)           ((base)->ISFR)

/*!
 * @}
 */
/* /* end of group PORT_Register_Accessor_Macros */
```

```
/*
-----  
-- PORT Register Masks  
----- */
```

```
/*!  
 * @addtogroup PORT_Register_Masks PORT Register Masks  
 * @{  
 */
```

```
/* PCR Bit Fields */

#define PORT_PCR_PS_MASK          0x1u
#define PORT_PCR_PS_SHIFT         0
#define PORT_PCR_PS_WIDTH         1
#define PORT_PCR_PS(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_PS_SHIFT))&PORT_PCR_PS_MASK

#define PORT_PCR_PE_MASK          0x2u
#define PORT_PCR_PE_SHIFT         1
#define PORT_PCR_PE_WIDTH         1
#define PORT_PCR_PE(x)            (((uint32_t)((uint32_t)(x))<<PORT_PCR_PE_SHIFT))&PORT_PCR_PE_MASK

#define PORT_PCR_SRE_MASK          0x4u
```

```

#define PORT_PCR_SRE_SHIFT          2
#define PORT_PCR_SRE_WIDTH          1
#define PORT_PCR_SRE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_SRE_SHIFT))&PORT_PCR_SRE_MASK)

#define PORT_PCR_PFE_MASK           0x10u
#define PORT_PCR_PFE_SHIFT          4
#define PORT_PCR_PFE_WIDTH          1
#define PORT_PCR_PFE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_PFE_SHIFT))&PORT_PCR_PFE_MASK)

#define PORT_PCR_DSE_MASK           0x40u
#define PORT_PCR_DSE_SHIFT          6
#define PORT_PCR_DSE_WIDTH          1
#define PORT_PCR_DSE(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_DSE_SHIFT))&PORT_PCR_DSE_MASK)

#define PORT_PCR_MUX_MASK           0x700u
#define PORT_PCR_MUX_SHIFT          8
#define PORT_PCR_MUX_WIDTH          3
#define PORT_PCR_MUX(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_MUX_SHIFT))&PORT_PCR_MUX_MASK)

#define PORT_PCR_IRQC_MASK           0xF0000u
#define PORT_PCR_IRQC_SHIFT          16
#define PORT_PCR_IRQC_WIDTH          4
#define PORT_PCR_IRQC(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_IRQC_SHIFT))&PORT_PCR_IRQC_MASK)

#define PORT_PCR_ISF_MASK            0x1000000u
#define PORT_PCR_ISF_SHIFT           24
#define PORT_PCR_ISF_WIDTH           1
#define PORT_PCR_ISF(x) (((uint32_t)((uint32_t)(x)<<PORT_PCR_ISF_SHIFT))&PORT_PCR_ISF_MASK)

/* GPCLR Bit Fields */

#define PORT_GPCLR_GPWD_MASK         0xFFFFu

```

```

#define PORT_GPCLR_GPWD_SHIFT          0
#define PORT_GPCLR_GPWD_WIDTH          16
#define PORT_GPCLR_GPWD(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWD_SHIFT))&PORT_GPCLR_GPWD_MASK)
#define PORT_GPCLR_GPWE_MASK           0xFFFF0000u
#define PORT_GPCLR_GPWE_SHIFT          16
#define PORT_GPCLR_GPWE_WIDTH          16
#define PORT_GPCLR_GPWE(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCLR_GPWE_SHIFT))&PORT_GPCLR_GPWE_MASK)

/* GPCHR Bit Fields */

#define PORT_GPCHR_GPWD_MASK           0xFFFFu
#define PORT_GPCHR_GPWD_SHIFT          0
#define PORT_GPCHR_GPWD_WIDTH          16
#define PORT_GPCHR_GPWD(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWD_SHIFT))&PORT_GPCHR_GPWD_MASK)
#define PORT_GPCHR_GPWE_MASK           0xFFFF0000u
#define PORT_GPCHR_GPWE_SHIFT          16
#define PORT_GPCHR_GPWE_WIDTH          16
#define PORT_GPCHR_GPWE(x)             (((uint32_t)((uint32_t)(x)<<PORT_GPCHR_GPWE_SHIFT))&PORT_GPCHR_GPWE_MASK)

/* ISFR Bit Fields */

#define PORT_ISFR_ISF_MASK             0xFFFFFFFFu
#define PORT_ISFR_ISF_SHIFT            0
#define PORT_ISFR_ISF_WIDTH            32
#define PORT_ISFR_ISF(x)               (((uint32_t)((uint32_t)(x)<<PORT_ISFR_ISF_SHIFT))&PORT_ISFR_ISF_MASK)

/*
 * @}
 */

/* /* end of group PORT_Register_Masks */

```

```
/* PORT - Peripheral instance base addresses */

/** Peripheral PORTA base address */

#define PORTA_BASE          (0x40049000u)

/** Peripheral PORTA base pointer */

#define PORTA          ((PORT_Type *)PORTA_BASE)

#define PORTA_BASE_PTR      (PORTA)

/** Peripheral PORTB base address */

#define PORTB_BASE          (0x4004A000u)

/** Peripheral PORTB base pointer */

#define PORTB          ((PORT_Type *)PORTB_BASE)

#define PORTB_BASE_PTR      (PORTB)

/** Peripheral PORTC base address */

#define PORTC_BASE          (0x4004B000u)

/** Peripheral PORTC base pointer */

#define PORTC          ((PORT_Type *)PORTC_BASE)

#define PORTC_BASE_PTR      (PORTC)

/** Peripheral PORTD base address */

#define PORTD_BASE          (0x4004C000u)

/** Peripheral PORTD base pointer */

#define PORTD          ((PORT_Type *)PORTD_BASE)

#define PORTD_BASE_PTR      (PORTD)

/** Peripheral PORTE base address */

#define PORTE_BASE          (0x4004D000u)

/** Peripheral PORTE base pointer */

#define PORTE          ((PORT_Type *)PORTE_BASE)

#define PORTE_BASE_PTR      (PORTE)

/** Array initializer of PORT peripheral base addresses */
```

```

#define PORT_BASE_ADDRS          { PORTA_BASE, PORTB_BASE, PORTC_BASE, PORTD_BASE,
PORTE_BASE }

/** Array initializer of PORT peripheral base pointers */

#define PORT_BASE_PTRS           { PORTA, PORTB, PORTC, PORTD, PORTE }

/*
-----*
-- PORT - Register accessor macros
----- */
/*!
* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor macros
* @{
*/
/* PORT - Register instance definitions */
/* PORTA */

#define PORTA_PCR0                PORT_PCR_REG(PORTA,0)
#define PORTA_PCR1                PORT_PCR_REG(PORTA,1)
#define PORTA_PCR2                PORT_PCR_REG(PORTA,2)
#define PORTA_PCR3                PORT_PCR_REG(PORTA,3)
#define PORTA_PCR4                PORT_PCR_REG(PORTA,4)
#define PORTA_PCR5                PORT_PCR_REG(PORTA,5)
#define PORTA_PCR6                PORT_PCR_REG(PORTA,6)
#define PORTA_PCR7                PORT_PCR_REG(PORTA,7)
#define PORTA_PCR8                PORT_PCR_REG(PORTA,8)
#define PORTA_PCR9                PORT_PCR_REG(PORTA,9)
#define PORTA_PCR10               PORT_PCR_REG(PORTA,10)
#define PORTA_PCR11               PORT_PCR_REG(PORTA,11)

```

```
#define PORTA_PCR12          PORT_PCR_REG(PORTA,12)
#define PORTA_PCR13          PORT_PCR_REG(PORTA,13)
#define PORTA_PCR14          PORT_PCR_REG(PORTA,14)
#define PORTA_PCR15          PORT_PCR_REG(PORTA,15)
#define PORTA_PCR16          PORT_PCR_REG(PORTA,16)
#define PORTA_PCR17          PORT_PCR_REG(PORTA,17)
#define PORTA_PCR18          PORT_PCR_REG(PORTA,18)
#define PORTA_PCR19          PORT_PCR_REG(PORTA,19)
#define PORTA_PCR20          PORT_PCR_REG(PORTA,20)
#define PORTA_PCR21          PORT_PCR_REG(PORTA,21)
#define PORTA_PCR22          PORT_PCR_REG(PORTA,22)
#define PORTA_PCR23          PORT_PCR_REG(PORTA,23)
#define PORTA_PCR24          PORT_PCR_REG(PORTA,24)
#define PORTA_PCR25          PORT_PCR_REG(PORTA,25)
#define PORTA_PCR26          PORT_PCR_REG(PORTA,26)
#define PORTA_PCR27          PORT_PCR_REG(PORTA,27)
#define PORTA_PCR28          PORT_PCR_REG(PORTA,28)
#define PORTA_PCR29          PORT_PCR_REG(PORTA,29)
#define PORTA_PCR30          PORT_PCR_REG(PORTA,30)
#define PORTA_PCR31          PORT_PCR_REG(PORTA,31)
#define PORTA_GPCLR          PORT_GPCLR_REG(PORTA)
#define PORTA_GPCHR          PORT_GPCHR_REG(PORTA)
#define PORTA_ISFR           PORT_ISFR_REG(PORTA)

/* PORTB */

#define PORTB_PCR0           PORT_PCR_REG(PORTB,0)
#define PORTB_PCR1           PORT_PCR_REG(PORTB,1)
#define PORTB_PCR2           PORT_PCR_REG(PORTB,2)
#define PORTB_PCR3           PORT_PCR_REG(PORTB,3)
#define PORTB_PCR4           PORT_PCR_REG(PORTB,4)
```

#define PORTB_PCR5	PORT_PCR_REG(PORTB,5)
#define PORTB_PCR6	PORT_PCR_REG(PORTB,6)
#define PORTB_PCR7	PORT_PCR_REG(PORTB,7)
#define PORTB_PCR8	PORT_PCR_REG(PORTB,8)
#define PORTB_PCR9	PORT_PCR_REG(PORTB,9)
#define PORTB_PCR10	PORT_PCR_REG(PORTB,10)
#define PORTB_PCR11	PORT_PCR_REG(PORTB,11)
#define PORTB_PCR12	PORT_PCR_REG(PORTB,12)
#define PORTB_PCR13	PORT_PCR_REG(PORTB,13)
#define PORTB_PCR14	PORT_PCR_REG(PORTB,14)
#define PORTB_PCR15	PORT_PCR_REG(PORTB,15)
#define PORTB_PCR16	PORT_PCR_REG(PORTB,16)
#define PORTB_PCR17	PORT_PCR_REG(PORTB,17)
#define PORTB_PCR18	PORT_PCR_REG(PORTB,18)
#define PORTB_PCR19	PORT_PCR_REG(PORTB,19)
#define PORTB_PCR20	PORT_PCR_REG(PORTB,20)
#define PORTB_PCR21	PORT_PCR_REG(PORTB,21)
#define PORTB_PCR22	PORT_PCR_REG(PORTB,22)
#define PORTB_PCR23	PORT_PCR_REG(PORTB,23)
#define PORTB_PCR24	PORT_PCR_REG(PORTB,24)
#define PORTB_PCR25	PORT_PCR_REG(PORTB,25)
#define PORTB_PCR26	PORT_PCR_REG(PORTB,26)
#define PORTB_PCR27	PORT_PCR_REG(PORTB,27)
#define PORTB_PCR28	PORT_PCR_REG(PORTB,28)
#define PORTB_PCR29	PORT_PCR_REG(PORTB,29)
#define PORTB_PCR30	PORT_PCR_REG(PORTB,30)
#define PORTB_PCR31	PORT_PCR_REG(PORTB,31)
#define PORTB_GPCLR	PORT_GPCLR_REG(PORTB)
#define PORTB_GPCHR	PORT_GPCHR_REG(PORTB)

```
#define PORTB_ISFR          PORT_ISFR_REG(PORTB)
/* PORTC */
#define PORTC_PCR0            PORT_PCR_REG(PORTC,0)
#define PORTC_PCR1            PORT_PCR_REG(PORTC,1)
#define PORTC_PCR2            PORT_PCR_REG(PORTC,2)
#define PORTC_PCR3            PORT_PCR_REG(PORTC,3)
#define PORTC_PCR4            PORT_PCR_REG(PORTC,4)
#define PORTC_PCR5            PORT_PCR_REG(PORTC,5)
#define PORTC_PCR6            PORT_PCR_REG(PORTC,6)
#define PORTC_PCR7            PORT_PCR_REG(PORTC,7)
#define PORTC_PCR8            PORT_PCR_REG(PORTC,8)
#define PORTC_PCR9            PORT_PCR_REG(PORTC,9)
#define PORTC_PCR10           PORT_PCR_REG(PORTC,10)
#define PORTC_PCR11           PORT_PCR_REG(PORTC,11)
#define PORTC_PCR12           PORT_PCR_REG(PORTC,12)
#define PORTC_PCR13           PORT_PCR_REG(PORTC,13)
#define PORTC_PCR14           PORT_PCR_REG(PORTC,14)
#define PORTC_PCR15           PORT_PCR_REG(PORTC,15)
#define PORTC_PCR16           PORT_PCR_REG(PORTC,16)
#define PORTC_PCR17           PORT_PCR_REG(PORTC,17)
#define PORTC_PCR18           PORT_PCR_REG(PORTC,18)
#define PORTC_PCR19           PORT_PCR_REG(PORTC,19)
#define PORTC_PCR20           PORT_PCR_REG(PORTC,20)
#define PORTC_PCR21           PORT_PCR_REG(PORTC,21)
#define PORTC_PCR22           PORT_PCR_REG(PORTC,22)
#define PORTC_PCR23           PORT_PCR_REG(PORTC,23)
#define PORTC_PCR24           PORT_PCR_REG(PORTC,24)
#define PORTC_PCR25           PORT_PCR_REG(PORTC,25)
#define PORTC_PCR26           PORT_PCR_REG(PORTC,26)
```

```
#define PORTC_PCR27          PORT_PCR_REG(PORTC,27)
#define PORTC_PCR28          PORT_PCR_REG(PORTC,28)
#define PORTC_PCR29          PORT_PCR_REG(PORTC,29)
#define PORTC_PCR30          PORT_PCR_REG(PORTC,30)
#define PORTC_PCR31          PORT_PCR_REG(PORTC,31)
#define PORTC_GPCLR          PORT_GPCLR_REG(PORTC)
#define PORTC_GPCHR          PORT_GPCHR_REG(PORTC)
#define PORTC_ISFR           PORT_ISFR_REG(PORTC)
/* PORTD */
#define PORTD_PCR0           PORT_PCR_REG(PORTD,0)
#define PORTD_PCR1           PORT_PCR_REG(PORTD,1)
#define PORTD_PCR2           PORT_PCR_REG(PORTD,2)
#define PORTD_PCR3           PORT_PCR_REG(PORTD,3)
#define PORTD_PCR4           PORT_PCR_REG(PORTD,4)
#define PORTD_PCR5           PORT_PCR_REG(PORTD,5)
#define PORTD_PCR6           PORT_PCR_REG(PORTD,6)
#define PORTD_PCR7           PORT_PCR_REG(PORTD,7)
#define PORTD_PCR8           PORT_PCR_REG(PORTD,8)
#define PORTD_PCR9           PORT_PCR_REG(PORTD,9)
#define PORTD_PCR10          PORT_PCR_REG(PORTD,10)
#define PORTD_PCR11          PORT_PCR_REG(PORTD,11)
#define PORTD_PCR12          PORT_PCR_REG(PORTD,12)
#define PORTD_PCR13          PORT_PCR_REG(PORTD,13)
#define PORTD_PCR14          PORT_PCR_REG(PORTD,14)
#define PORTD_PCR15          PORT_PCR_REG(PORTD,15)
#define PORTD_PCR16          PORT_PCR_REG(PORTD,16)
#define PORTD_PCR17          PORT_PCR_REG(PORTD,17)
#define PORTD_PCR18          PORT_PCR_REG(PORTD,18)
#define PORTD_PCR19          PORT_PCR_REG(PORTD,19)
```

```

#define PORTD_PCR20      PORT_PCR_REG(PORTD,20)
#define PORTD_PCR21      PORT_PCR_REG(PORTD,21)
#define PORTD_PCR22      PORT_PCR_REG(PORTD,22)
#define PORTD_PCR23      PORT_PCR_REG(PORTD,23)
#define PORTD_PCR24      PORT_PCR_REG(PORTD,24)
#define PORTD_PCR25      PORT_PCR_REG(PORTD,25)
#define PORTD_PCR26      PORT_PCR_REG(PORTD,26)
#define PORTD_PCR27      PORT_PCR_REG(PORTD,27)
#define PORTD_PCR28      PORT_PCR_REG(PORTD,28)
#define PORTD_PCR29      PORT_PCR_REG(PORTD,29)
#define PORTD_PCR30      PORT_PCR_REG(PORTD,30)
#define PORTD_PCR31      PORT_PCR_REG(PORTD,31)
#define PORTD_GPCLR      PORT_GPCLR_REG(PORTD)
#define PORTD_GPCHR      PORT_GPCHR_REG(PORTD)
#define PORTD_ISFR       PORT_ISFR_REG(PORTD)

/* PORTE */

#define PORTE_PCR0       PORT_PCR_REG(PORTE,0)
#define PORTE_PCR1       PORT_PCR_REG(PORTE,1)
#define PORTE_PCR2       PORT_PCR_REG(PORTE,2)
#define PORTE_PCR3       PORT_PCR_REG(PORTE,3)
#define PORTE_PCR4       PORT_PCR_REG(PORTE,4)
#define PORTE_PCR5       PORT_PCR_REG(PORTE,5)
#define PORTE_PCR6       PORT_PCR_REG(PORTE,6)
#define PORTE_PCR7       PORT_PCR_REG(PORTE,7)
#define PORTE_PCR8       PORT_PCR_REG(PORTE,8)
#define PORTE_PCR9       PORT_PCR_REG(PORTE,9)
#define PORTE_PCR10      PORT_PCR_REG(PORTE,10)
#define PORTE_PCR11      PORT_PCR_REG(PORTE,11)
#define PORTE_PCR12      PORT_PCR_REG(PORTE,12)

```

#define PORTE_PCR13	PORT_PCR_REG(PORTE,13)
#define PORTE_PCR14	PORT_PCR_REG(PORTE,14)
#define PORTE_PCR15	PORT_PCR_REG(PORTE,15)
#define PORTE_PCR16	PORT_PCR_REG(PORTE,16)
#define PORTE_PCR17	PORT_PCR_REG(PORTE,17)
#define PORTE_PCR18	PORT_PCR_REG(PORTE,18)
#define PORTE_PCR19	PORT_PCR_REG(PORTE,19)
#define PORTE_PCR20	PORT_PCR_REG(PORTE,20)
#define PORTE_PCR21	PORT_PCR_REG(PORTE,21)
#define PORTE_PCR22	PORT_PCR_REG(PORTE,22)
#define PORTE_PCR23	PORT_PCR_REG(PORTE,23)
#define PORTE_PCR24	PORT_PCR_REG(PORTE,24)
#define PORTE_PCR25	PORT_PCR_REG(PORTE,25)
#define PORTE_PCR26	PORT_PCR_REG(PORTE,26)
#define PORTE_PCR27	PORT_PCR_REG(PORTE,27)
#define PORTE_PCR28	PORT_PCR_REG(PORTE,28)
#define PORTE_PCR29	PORT_PCR_REG(PORTE,29)
#define PORTE_PCR30	PORT_PCR_REG(PORTE,30)
#define PORTE_PCR31	PORT_PCR_REG(PORTE,31)
#define PORTE_GPCLR	PORT_GPCLR_REG(PORTE)
#define PORTE_GPCHR	PORT_GPCHR_REG(PORTE)
#define PORTE_ISFR	PORT_ISFR_REG(PORTE)

/* PORT - Register array accessors */

#define PORTA_PCR(index)	PORT_PCR_REG(PORTA,index)
#define PORTB_PCR(index)	PORT_PCR_REG(PORTB,index)
#define PORTC_PCR(index)	PORT_PCR_REG(PORTC,index)
#define PORTD_PCR(index)	PORT_PCR_REG(PORTD,index)
#define PORTE_PCR(index)	PORT_PCR_REG(PORTE,index)

```
/*!  
 * @}  
 */ /* end of group PORT_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group PORT_Peripheral_Access_Layer */
```

```
/* -----  
-- RCM Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup RCM_Peripheral_Access_Layer RCM Peripheral Access Layer  
 * @{  
 */
```

```
/** RCM - Register Layout Typedef */  
  
typedef struct {  
    __I uint8_t SRS0;           /**< System Reset Status Register 0, offset: 0x0 */  
    __I uint8_t SRS1;           /**< System Reset Status Register 1, offset: 0x1 */  
    uint8_t RESERVED_0[2];  
    __IO uint8_t RPFC;          /**< Reset Pin Filter Control register, offset: 0x4 */  
    __IO uint8_t RPFW;          /**< Reset Pin Filter Width register, offset: 0x5 */  
} RCM_Type, *RCM_MemMapPtr;
```

```
/* -----
-- RCM - Register accessor macros
----- */

/*!
* @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros
* @{
*
/* RCM - Register accessors */
#define RCM_SRS0_REG(base) ((base)->SRS0)
#define RCM_SRS1_REG(base) ((base)->SRS1)
#define RCM_RPFC_REG(base) ((base)->RPFC)
#define RCM_RPFW_REG(base) ((base)->RPFW)

/*!
* @}
*/
/* end of group RCM_Register_Accessor_Macros */
```

```
/* -----
-- RCM Register Masks
----- */

/*!
* @addtogroup RCM_Register_Masks RCM Register Masks
* @{
*
*/
```

```

/* SRS0 Bit Fields */

#define RCM_SRS0_WAKEUP_MASK          0x1u
#define RCM_SRS0_WAKEUP_SHIFT         0
#define RCM_SRS0_WAKEUP_WIDTH         1
#define RCM_SRS0_WAKEUP(x)            (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WAKEUP_SHIFT))&RCM_SRS0_WAKEUP_MASK)

#define RCM_SRS0_LVD_MASK            0x2u
#define RCM_SRS0_LVD_SHIFT           1
#define RCM_SRS0_LVD_WIDTH           1
#define RCM_SRS0_LVD(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LVD_SHIFT))&RCM_SRS0_LVD_MASK)

#define RCM_SRS0_LOC_MASK            0x4u
#define RCM_SRS0_LOC_SHIFT           2
#define RCM_SRS0_LOC_WIDTH           1
#define RCM_SRS0_LOC(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOC_SHIFT))&RCM_SRS0_LOC_MASK)

#define RCM_SRS0_LOL_MASK            0x8u
#define RCM_SRS0_LOL_SHIFT           3
#define RCM_SRS0_LOL_WIDTH           1
#define RCM_SRS0_LOL(x)              (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOL_SHIFT))&RCM_SRS0_LOL_MASK)

#define RCM_SRS0_WDOG_MASK           0x20u
#define RCM_SRS0_WDOG_SHIFT          5
#define RCM_SRS0_WDOG_WIDTH          1
#define RCM_SRS0_WDOG(x)             (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WDOG_SHIFT))&RCM_SRS0_WDOG_MASK)

#define RCM_SRS0_PIN_MASK            0x40u
#define RCM_SRS0_PIN_SHIFT           6
#define RCM_SRS0_PIN_WIDTH           1

```

```

#define RCM_SRS0_PIN(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS0_PIN_SHIFT))&RCM_SRS0_PIN_MASK)

#define RCM_SRS0_POR_MASK          0x80u
#define RCM_SRS0_POR_SHIFT         7
#define RCM_SRS0_POR_WIDTH         1

#define RCM_SRS0_POR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS0_POR_SHIFT))&RCM_SRS0_POR_MASK)

/* SRS1 Bit Fields */

#define RCM_SRS1_LOCKUP_MASK       0x2u
#define RCM_SRS1_LOCKUP_SHIFT      1
#define RCM_SRS1_LOCKUP_WIDTH      1

#define RCM_SRS1_LOCKUP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_LOCKUP_SHIFT))&RCM_SRS1_LOCKUP_MASK)

#define RCM_SRS1_SW_MASK           0x4u
#define RCM_SRS1_SW_SHIFT          2
#define RCM_SRS1_SW_WIDTH          1

#define RCM_SRS1_SW(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SW_SHIFT))&RCM_SRS1_SW_MASK)

#define RCM_SRS1_MDM_AP_MASK        0x8u
#define RCM_SRS1_MDM_AP_SHIFT       3
#define RCM_SRS1_MDM_AP_WIDTH       1

#define RCM_SRS1_MDM_AP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_MDM_AP_SHIFT))&RCM_SRS1_MDM_AP_MASK)

#define RCM_SRS1_SACKERR_MASK       0x20u
#define RCM_SRS1_SACKERR_SHIFT      5
#define RCM_SRS1_SACKERR_WIDTH      1

#define RCM_SRS1_SACKERR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SACKERR_SHIFT))&RCM_SRS1_SACKERR_MASK)

/* RPFC Bit Fields */

#define RCM_RPFC_RSTFLTSRW_MASK     0x3u
#define RCM_RPFC_RSTFLTSRW_SHIFT     0

```

```

#define RCM_RPFC_RSTFLTSRW_WIDTH          2
#define RCM_RPFC_RSTFLTSRW(x) (((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSRW_SHIFT))&RCM_RPFC_RSTFLTSRW_MASK)
#define RCM_RPFC_RSTFLTSS_MASK            0x4u
#define RCM_RPFC_RSTFLTSS_SHIFT          2
#define RCM_RPFC_RSTFLTSS_WIDTH          1
#define RCM_RPFC_RSTFLTSS(x) (((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSS_SHIFT))&RCM_RPFC_RSTFLTSS_MASK)

/* RPFW Bit Fields */

#define RCM_RPFW_RSTFLTSEL_MASK          0x1Fu
#define RCM_RPFW_RSTFLTSEL_SHIFT         0
#define RCM_RPFW_RSTFLTSEL_WIDTH         5
#define RCM_RPFW_RSTFLTSEL(x) (((uint8_t)((uint8_t)(x)<<RCM_RPFW_RSTFLTSEL_SHIFT))&RCM_RPFW_RSTFLTSEL_MASK)

/*
 */

```

```

/* RCM - Peripheral instance base addresses */

/** Peripheral RCM base address */
#define RCM_BASE                      (0x4007F000u)

/** Peripheral RCM base pointer */
#define RCM                         ((RCM_Type *)RCM_BASE)

#define RCM_BASE_PTR                  (RCM)

/** Array initializer of RCM peripheral base addresses */
#define RCM_BASE_ADDRS                { RCM_BASE }

/** Array initializer of RCM peripheral base pointers */
#define RCM_BASE_PTRS                 { RCM }

```

```
/*
-----  
-- RCM - Register accessor macros  
----- */  
  
/*!  
 * @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor macros  
 * @{  
 */  
  
/* RCM - Register instance definitions */  
/* RCM */  
  
#define RCM_SRS0          RCM_SRS0_REG(RCM)  
#define RCM_SRS1          RCM_SRS1_REG(RCM)  
#define RCM_RPFC          RCM_RPFC_REG(RCM)  
#define RCM_RPFW          RCM_RPFW_REG(RCM)  
  
/*!  
 * @}  
 */ /* end of group RCM_Register_Accessor_Macros */  
  
/*!  
 * @}  
 */ /* end of group RCM_Peripheral_Access_Layer */  
  
/*
-----
```

```

-- ROM Peripheral Access Layer
----- */
/*!
* @addtogroup ROM_Peripheral_Access_Layer ROM Peripheral Access Layer
* @{
*/
/** ROM - Register Layout Typedef */
typedef struct {
    __I uint32_t ENTRY[3];           /**< Entry, array offset: 0x0, array step: 0x4 */
    __I uint32_t TABLEMARK;          /**< End of Table Marker Register, offset: 0xC */
    uint8_t RESERVED_0[4028];
    __I uint32_t SYSACCESS;          /**< System Access Register, offset: 0xFCC */
    __I uint32_t PERIPHID4;          /**< Peripheral ID Register, offset: 0xFD0 */
    __I uint32_t PERIPHID5;          /**< Peripheral ID Register, offset: 0xFD4 */
    __I uint32_t PERIPHID6;          /**< Peripheral ID Register, offset: 0xFD8 */
    __I uint32_t PERIPHID7;          /**< Peripheral ID Register, offset: 0xFDC */
    __I uint32_t PERIPHID0;          /**< Peripheral ID Register, offset: 0xFE0 */
    __I uint32_t PERIPHID1;          /**< Peripheral ID Register, offset: 0xFE4 */
    __I uint32_t PERIPHID2;          /**< Peripheral ID Register, offset: 0xFE8 */
    __I uint32_t PERIPHID3;          /**< Peripheral ID Register, offset: 0xFEC */
    __I uint32_t COMPID[4];          /**< Component ID Register, array offset: 0xFF0, array step: 0x4
*/
} ROM_Type, *ROM_MemMapPtr;

```

```

/*
----- */
-- ROM - Register accessor macros
----- */

```

```

/*!

* @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros

* @{

*/



/* ROM - Register accessors */

#define ROM_ENTRY_REG(base,index)      ((base)->ENTRY[index])

#define ROM_ENTRY_COUNT           3

#define ROM_TABLEMARK_REG(base)     ((base)->TABLEMARK)

#define ROM_SYSACCESS_REG(base)    ((base)->SYSACCESS)

#define ROM_PERIPHID4_REG(base)   ((base)->PERIPHID4)

#define ROM_PERIPHID5_REG(base)   ((base)->PERIPHID5)

#define ROM_PERIPHID6_REG(base)   ((base)->PERIPHID6)

#define ROM_PERIPHID7_REG(base)   ((base)->PERIPHID7)

#define ROM_PERIPHID0_REG(base)   ((base)->PERIPHID0)

#define ROM_PERIPHID1_REG(base)   ((base)->PERIPHID1)

#define ROM_PERIPHID2_REG(base)   ((base)->PERIPHID2)

#define ROM_PERIPHID3_REG(base)   ((base)->PERIPHID3)

#define ROM_COMPID_REG(base,index) ((base)->COMPID[index])

#define ROM_COMPID_COUNT          4


/*!


* @}

*/ /* end of group ROM_Register_Accessor_Macros */

```

```

/* -----

```

```

-- ROM Register Masks
----- */
/*!
* @addtogroup ROM_Register_Masks ROM Register Masks
* @{
*/
/* ENTRY Bit Fields */

#define ROM_ENTRY_ENTRY_MASK          0xFFFFFFFFu
#define ROM_ENTRY_ENTRY_SHIFT         0
#define ROM_ENTRY_ENTRY_WIDTH         32
#define ROM_ENTRY_ENTRY(x)            (((uint32_t)((uint32_t)(x))<<ROM_ENTRY_ENTRY_SHIFT))&ROM_ENTRY_ENTRY_MASK

/* TABLEMARK Bit Fields */

#define ROM_TABLEMARK_MARK_MASK       0xFFFFFFFFu
#define ROM_TABLEMARK_MARK_SHIFT      0
#define ROM_TABLEMARK_MARK_WIDTH      32
#define ROM_TABLEMARK_MARK(x)         (((uint32_t)((uint32_t)(x))<<ROM_TABLEMARK_MARK_SHIFT))&ROM_TABLEMARK_MARK_MASK

/* SYSACCESS Bit Fields */

#define ROM_SYSACCESS_SYSACCESS_MASK  0xFFFFFFFFu
#define ROM_SYSACCESS_SYSACCESS_SHIFT 0
#define ROM_SYSACCESS_SYSACCESS_WIDTH 32
#define ROM_SYSACCESS_SYSACCESS(x)    (((uint32_t)((uint32_t)(x))<<ROM_SYSACCESS_SYSACCESS_SHIFT))&ROM_SYSACCESS_SYSACCESS_MASK

/* PERIPHID4 Bit Fields */

#define ROM_PERIPHID4_PERIPHID_MASK   0xFFFFFFFFu
#define ROM_PERIPHID4_PERIPHID_SHIFT  0

```

```

#define ROM_PERIPHID4_PERIPHID_WIDTH      32
#define ROM_PERIPHID4_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID4_PERIPHID_SHIFT))&ROM_PERIPHID4_PERIPHID_MASK)

/* PERIPHID5 Bit Fields */

#define ROM_PERIPHID5_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID5_PERIPHID_SHIFT       0
#define ROM_PERIPHID5_PERIPHID_WIDTH       32
#define ROM_PERIPHID5_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID5_PERIPHID_SHIFT))&ROM_PERIPHID5_PERIPHID_MASK)

/* PERIPHID6 Bit Fields */

#define ROM_PERIPHID6_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID6_PERIPHID_SHIFT       0
#define ROM_PERIPHID6_PERIPHID_WIDTH       32
#define ROM_PERIPHID6_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID6_PERIPHID_SHIFT))&ROM_PERIPHID6_PERIPHID_MASK)

/* PERIPHID7 Bit Fields */

#define ROM_PERIPHID7_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID7_PERIPHID_SHIFT       0
#define ROM_PERIPHID7_PERIPHID_WIDTH       32
#define ROM_PERIPHID7_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID7_PERIPHID_SHIFT))&ROM_PERIPHID7_PERIPHID_MASK)

/* PERIPHID0 Bit Fields */

#define ROM_PERIPHIDO_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHIDO_PERIPHID_SHIFT       0
#define ROM_PERIPHIDO_PERIPHID_WIDTH       32
#define ROM_PERIPHIDO_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHIDO_PERIPHID_SHIFT))&ROM_PERIPHIDO_PERIPHID_MASK)

/* PERIPHID1 Bit Fields */

#define ROM_PERIPHID1_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID1_PERIPHID_SHIFT       0

```

```

#define ROM_PERIPHID1_PERIPHID_WIDTH      32
#define ROM_PERIPHID1_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID1_PERIPHID_SHIFT))&ROM_PERIPHID1_PERIPHID_MASK)

/* PERIPHID2 Bit Fields */

#define ROM_PERIPHID2_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID2_PERIPHID_SHIFT       0
#define ROM_PERIPHID2_PERIPHID_WIDTH       32
#define ROM_PERIPHID2_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID2_PERIPHID_SHIFT))&ROM_PERIPHID2_PERIPHID_MASK)

/* PERIPHID3 Bit Fields */

#define ROM_PERIPHID3_PERIPHID_MASK        0xFFFFFFFFu
#define ROM_PERIPHID3_PERIPHID_SHIFT       0
#define ROM_PERIPHID3_PERIPHID_WIDTH       32
#define ROM_PERIPHID3_PERIPHID(x)
(((uint32_t)((uint32_t)(x)<<ROM_PERIPHID3_PERIPHID_SHIFT))&ROM_PERIPHID3_PERIPHID_MASK)

/* COMPID Bit Fields */

#define ROM_COMPID_COMPID_MASK            0xFFFFFFFFu
#define ROM_COMPID_COMPID_SHIFT          0
#define ROM_COMPID_COMPID_WIDTH          32
#define ROM_COMPID_COMPID(x)
(((uint32_t)((uint32_t)(x)<<ROM_COMPID_COMPID_SHIFT))&ROM_COMPID_COMPID_MASK)

/*!
 * @}
 */

/* /* end of group ROM_Register_Masks */

```

```

/* ROM - Peripheral instance base addresses */
/** Peripheral ROM base address */
#define ROM_BASE                      (0xF0002000u)

```

```

/** Peripheral ROM base pointer */

#define ROM           ((ROM_Type *)ROM_BASE)

#define ROM_BASE_PTR      (ROM)

/** Array initializer of ROM peripheral base addresses */

#define ROM_BASE_ADDRS    { ROM_BASE }

/** Array initializer of ROM peripheral base pointers */

#define ROM_BASE_PTRS     { ROM }

/*
-----*
-- ROM - Register accessor macros
----- */
/*!
* @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor macros
* @{
*/
/* ROM - Register instance definitions */

/* ROM */

#define ROM_ENTRY0        ROM_ENTRY_REG(ROM,0)
#define ROM_ENTRY1        ROM_ENTRY_REG(ROM,1)
#define ROM_ENTRY2        ROM_ENTRY_REG(ROM,2)
#define ROM_TABLEMARK     ROM_TABLEMARK_REG(ROM)
#define ROM_SYSACCESS     ROM_SYSACCESS_REG(ROM)
#define ROM_PERIPHID4    ROM_PERIPHID4_REG(ROM)
#define ROM_PERIPHID5    ROM_PERIPHID5_REG(ROM)
#define ROM_PERIPHID6    ROM_PERIPHID6_REG(ROM)
#define ROM_PERIPHID7    ROM_PERIPHID7_REG(ROM)

```

```
#define ROM_PERIPHID0      ROM_PERIPHID0_REG(ROM)
#define ROM_PERIPHID1      ROM_PERIPHID1_REG(ROM)
#define ROM_PERIPHID2      ROM_PERIPHID2_REG(ROM)
#define ROM_PERIPHID3      ROM_PERIPHID3_REG(ROM)
#define ROM_COMPID0          ROM_COMPID_REG(ROM,0)
#define ROM_COMPID1          ROM_COMPID_REG(ROM,1)
#define ROM_COMPID2          ROM_COMPID_REG(ROM,2)
#define ROM_COMPID3          ROM_COMPID_REG(ROM,3)
```

```
/* ROM - Register array accessors */
```

```
#define ROM_ENTRY(index)    ROM_ENTRY_REG(ROM,index)
#define ROM_COMPID(index)    ROM_COMPID_REG(ROM,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group ROM_Register_Accessor_Macros */
```

```
/*!
```

```
* @}
```

```
/* /* end of group ROM_Peripheral_Access_Layer */
```

```
/* -----
```

```
-- RTC Peripheral Access Layer
```

```
----- */
```

```
/*!
```

```
* @addtogroup RTC_Peripheral_Access_Layer RTC Peripheral Access Layer
```

```

* @@
*/

```

/** RTC - Register Layout Typedef */

```

typedef struct {
    __IO uint32_t TSR;           /**< RTC Time Seconds Register, offset: 0x0 */
    __IO uint32_t TPR;           /**< RTC Time Prescaler Register, offset: 0x4 */
    __IO uint32_t TAR;           /**< RTC Time Alarm Register, offset: 0x8 */
    __IO uint32_t TCR;           /**< RTC Time Compensation Register, offset: 0xC */
    __IO uint32_t CR;            /**< RTC Control Register, offset: 0x10 */
    __IO uint32_t SR;            /**< RTC Status Register, offset: 0x14 */
    __IO uint32_t LR;            /**< RTC Lock Register, offset: 0x18 */
    __IO uint32_t IER;           /**< RTC Interrupt Enable Register, offset: 0x1C */
}
RTC_Type, *RTC_MemMapPtr;

```

```

/*
-----  

-- RTC - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros  

* @@
*/

```

```

/* RTC - Register accessors */  

#define RTC_TSREG(base)      ((base)->TSR)  

#define RTC_TPREG(base)       ((base)->TPR)  

#define RTC_TARREG(base)      ((base)->TAR)

```

```
#define RTC_TCR_REG(base) ((base)->TCR)
#define RTC_CR_REG(base) ((base)->CR)
#define RTC_SR_REG(base) ((base)->SR)
#define RTC_LR_REG(base) ((base)->LR)
#define RTC_IER_REG(base) ((base)->IER)
```

```
/*!
 * @}
 */
/* /* end of group RTC_Register_Accessor_Macros */
```

```
/*
-----  
-- RTC Register Masks  
----- */
```

```
/*!
 * @addtogroup RTC_Register_Masks RTC Register Masks
 * @{
 */

```

```
/* TSR Bit Fields */

#define RTC_TSRR_MASK 0xFFFFFFFFu
#define RTC_TSRR_SHIFT 0
#define RTC_TSRR_WIDTH 32

#define RTC_TSRR(x) (((uint32_t)((uint32_t)(x))<<RTC_TSRR_SHIFT))&RTC_TSRR_MASK

/* TPR Bit Fields */

#define RTC_TPR_TPR_MASK 0xFFFFu
#define RTC_TPR_TPR_SHIFT 0
```

```

#define RTC_TPR_TPR_WIDTH          16
#define RTC_TPR_TPR(x) (((uint32_t)((uint32_t)(x))<<RTC_TPR_TPR_SHIFT))&RTC_TPR_TPR_MASK)
/* TAR Bit Fields */

#define RTC_TAR_TAR_MASK           0xFFFFFFFFu
#define RTC_TAR_TAR_SHIFT          0
#define RTC_TAR_TAR_WIDTH          32
#define RTC_TAR_TAR(x) (((uint32_t)((uint32_t)(x))<<RTC_TAR_TAR_SHIFT))&RTC_TAR_TAR_MASK)

/* TCR Bit Fields */

#define RTC_TCR_TCR_MASK           0xFFu
#define RTC_TCR_TCR_SHIFT          0
#define RTC_TCR_TCR_WIDTH          8
#define RTC_TCR_TCR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCR_SHIFT))&RTC_TCR_TCR_MASK)

#define RTC_TCR_CIR_MASK           0xFF00u
#define RTC_TCR_CIR_SHIFT          8
#define RTC_TCR_CIR_WIDTH          8
#define RTC_TCR_CIR(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIR_SHIFT))&RTC_TCR_CIR_MASK)

#define RTC_TCR_TCV_MASK           0xFF0000u
#define RTC_TCR_TCV_SHIFT          16
#define RTC_TCR_TCV_WIDTH          8
#define RTC_TCR_TCV(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_TCV_SHIFT))&RTC_TCR_TCV_MASK)

#define RTC_TCR_CIC_MASK           0xFF000000u
#define RTC_TCR_CIC_SHIFT          24
#define RTC_TCR_CIC_WIDTH          8
#define RTC_TCR_CIC(x) (((uint32_t)((uint32_t)(x))<<RTC_TCR_CIC_SHIFT))&RTC_TCR_CIC_MASK)

/* CR Bit Fields */

```

```

#define RTC_CR_SWR_MASK           0x1u
#define RTC_CR_SWR_SHIFT          0
#define RTC_CR_SWR_WIDTH          1
#define RTC_CR_SWR(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_SWR_SHIFT))&RTC_CR_SWR_MASK)
#define RTC_CR_WPE_MASK            0x2u
#define RTC_CR_WPE_SHIFT           1
#define RTC_CR_WPE_WIDTH           1
#define RTC_CR_WPE(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_WPE_SHIFT))&RTC_CR_WPE_MASK)
#define RTC_CR_SUP_MASK             0x4u
#define RTC_CR_SUP_SHIFT            2
#define RTC_CR_SUP_WIDTH            1
#define RTC_CR_SUP(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_SUP_SHIFT))&RTC_CR_SUP_MASK)
#define RTC_CR_UM_MASK              0x8u
#define RTC_CR_UM_SHIFT              3
#define RTC_CR_UM_WIDTH              1
#define RTC_CR_UM(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_UM_SHIFT))&RTC_CR_UM_MASK)
#define RTC_CR_OSCE_MASK             0x100u
#define RTC_CR_OSCE_SHIFT             8
#define RTC_CR_OSCE_WIDTH             1
#define RTC_CR_OSCE(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_OSCE_SHIFT))&RTC_CR_OSCE_MASK)
#define RTC_CR_CLKO_MASK             0x200u
#define RTC_CR_CLKO_SHIFT             9
#define RTC_CR_CLKO_WIDTH             1
#define RTC_CR_CLKO(x) (((uint32_t)((uint32_t)(x))<<RTC_CR_CLKO_SHIFT))&RTC_CR_CLKO_MASK)
#define RTC_CR_SC16P_MASK             0x400u

```

```

#define RTC_CR_SC16P_SHIFT          10
#define RTC_CR_SC16P_WIDTH          1
#define RTC_CR_SC16P(x)             (((uint32_t)((uint32_t)(x))<<RTC_CR_SC16P_SHIFT))&RTC_CR_SC16P_MASK)
#define RTC_CR_SC8P_MASK            0x800u
#define RTC_CR_SC8P_SHIFT           11
#define RTC_CR_SC8P_WIDTH           1
#define RTC_CR_SC8P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC8P_SHIFT))&RTC_CR_SC8P_MASK)
#define RTC_CR_SC4P_MASK            0x1000u
#define RTC_CR_SC4P_SHIFT           12
#define RTC_CR_SC4P_WIDTH           1
#define RTC_CR_SC4P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC4P_SHIFT))&RTC_CR_SC4P_MASK)
#define RTC_CR_SC2P_MASK            0x2000u
#define RTC_CR_SC2P_SHIFT           13
#define RTC_CR_SC2P_WIDTH           1
#define RTC_CR_SC2P(x)              (((uint32_t)((uint32_t)(x))<<RTC_CR_SC2P_SHIFT))&RTC_CR_SC2P_MASK)

/* SR Bit Fields */

#define RTC_SR_TIF_MASK             0x1u
#define RTC_SR_TIF_SHIFT             0
#define RTC_SR_TIF_WIDTH             1
#define RTC_SR_TIF(x)                (((uint32_t)((uint32_t)(x))<<RTC_SR_TIF_SHIFT))&RTC_SR_TIF_MASK)
#define RTC_SR_TOF_MASK              0x2u
#define RTC_SR_TOF_SHIFT              1
#define RTC_SR_TOF_WIDTH              1
#define RTC_SR_TOF(x)                 (((uint32_t)((uint32_t)(x))<<RTC_SR_TOF_SHIFT))&RTC_SR_TOF_MASK)
#define RTC_SR_TAF_MASK              0x4u

```

```

#define RTC_SR_TAF_SHIFT          2
#define RTC_SR_TAF_WIDTH          1
#define RTC_SR_TAF(x)
(((uint32_t)((uint32_t)(x)<<RTC_SR_TAF_SHIFT))&RTC_SR_TAF_MASK)

#define RTC_SR_TCE_MASK           0x10u
#define RTC_SR_TCE_SHIFT          4
#define RTC_SR_TCE_WIDTH          1
#define RTC_SR_TCE(x)
(((uint32_t)((uint32_t)(x)<<RTC_SR_TCE_SHIFT))&RTC_SR_TCE_MASK)

/* LR Bit Fields */

#define RTC_LR_TCL_MASK           0x8u
#define RTC_LR_TCL_SHIFT          3
#define RTC_LR_TCL_WIDTH          1
#define RTC_LR_TCL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_TCL_SHIFT))&RTC_LR_TCL_MASK)

#define RTC_LR_CRL_MASK           0x10u
#define RTC_LR_CRL_SHIFT          4
#define RTC_LR_CRL_WIDTH          1
#define RTC_LR_CRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_CRL_SHIFT))&RTC_LR_CRL_MASK)

#define RTC_LR_SRL_MASK           0x20u
#define RTC_LR_SRL_SHIFT          5
#define RTC_LR_SRL_WIDTH          1
#define RTC_LR_SRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_SRL_SHIFT))&RTC_LR_SRL_MASK)

#define RTC_LR_LRL_MASK           0x40u
#define RTC_LR_LRL_SHIFT          6
#define RTC_LR_LRL_WIDTH          1
#define RTC_LR_LRL(x)
(((uint32_t)((uint32_t)(x)<<RTC_LR_LRL_SHIFT))&RTC_LR_LRL_MASK)

/* IER Bit Fields */

```

```

#define RTC_IER_TIIE_MASK           0x1u
#define RTC_IER_TIIE_SHIFT          0
#define RTC_IER_TIIE_WIDTH          1
#define RTC_IER_TIIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TIIE_SHIFT))&RTC_IER_TIIE_MASK

#define RTC_IER_TOIE_MASK           0x2u
#define RTC_IER_TOIE_SHIFT          1
#define RTC_IER_TOIE_WIDTH          1
#define RTC_IER_TOIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TOIE_SHIFT))&RTC_IER_TOIE_MASK

#define RTC_IER_TAIE_MASK           0x4u
#define RTC_IER_TAIE_SHIFT          2
#define RTC_IER_TAIE_WIDTH          1
#define RTC_IER_TAIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TAIE_SHIFT))&RTC_IER_TAIE_MASK

#define RTC_IER_TSIE_MASK           0x10u
#define RTC_IER_TSIE_SHIFT          4
#define RTC_IER_TSIE_WIDTH          1
#define RTC_IER_TSIE(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_TSIE_SHIFT))&RTC_IER_TSIE_MASK

#define RTC_IER_WPON_MASK           0x80u
#define RTC_IER_WPON_SHIFT          7
#define RTC_IER_WPON_WIDTH          1
#define RTC_IER_WPON(x)             (((uint32_t)((uint32_t)(x))<<RTC_IER_WPON_SHIFT))&RTC_IER_WPON_MASK

/*
 * @}
 */ /* end of group RTC_Register_Masks */

```

```

/* RTC - Peripheral instance base addresses */

/** Peripheral RTC base address */

#define RTC_BASE           (0x4003D000u)

/** Peripheral RTC base pointer */

#define RTC             ((RTC_Type *)RTC_BASE)

#define RTC_BASE_PTR      (RTC)

/** Array initializer of RTC peripheral base addresses */

#define RTC_BASE_ADDRS    { RTC_BASE }

/** Array initializer of RTC peripheral base pointers */

#define RTC_BASE_PTRS     { RTC }

/*
-----*
-- RTC - Register accessor macros
----- */
/*!
* @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor macros
* @{
*/
/* RTC - Register instance definitions */

/* RTC */

#define RTC_TSR           RTC_TSREG(RTC)
#define RTC_TPR           RTC_TSPRREG(RTC)
#define RTC_TAR           RTC_TARREG(RTC)
#define RTC_TCR           RTC_TCRRREG(RTC)
#define RTC_CR            RTC_CRRREG(RTC)

```

```
#define RTC_SR           RTC_SR_REG(RTC)
#define RTC_LR           RTC_LR_REG(RTC)
#define RTC_IER           RTC_IER_REG(RTC)
```

```
/*!
 * @}
 */
/*/* end of group RTC_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/*/* end of group RTC_Peripheral_Access_Layer */
```

```
/*
-----  
-- SIM Peripheral Access Layer  
----- */
```

```
/*!
 * @addtogroup SIM_Peripheral_Access_Layer SIM Peripheral Access Layer
 * @{
 */

```

```
/** SIM - Register Layout Typedef */
typedef struct {
    __IO uint32_t SOPT1;           /**< System Options Register 1, offset: 0x0 */
    __IO uint32_t SOPT1CFG;        /**< SOPT1 Configuration Register, offset: 0x4 */
    uint8_t RESERVED_0[4092];
    __IO uint32_t SOPT2;           /**< System Options Register 2, offset: 0x1004 */
}
```

```

    uint8_t RESERVED_1[4];

    __IO uint32_t SOPT4;           /**< System Options Register 4, offset: 0x100C */
    __IO uint32_t SOPT5;           /**< System Options Register 5, offset: 0x1010 */
    uint8_t RESERVED_2[4];
    __IO uint32_t SOPT7;           /**< System Options Register 7, offset: 0x1018 */
    uint8_t RESERVED_3[8];
    __I uint32_t SDID;            /**< System Device Identification Register, offset: 0x1024 */
    uint8_t RESERVED_4[12];
    __IO uint32_t SCGC4;           /**< System Clock Gating Control Register 4, offset: 0x1034 */
    __IO uint32_t SCGC5;           /**< System Clock Gating Control Register 5, offset: 0x1038 */
    __IO uint32_t SCGC6;           /**< System Clock Gating Control Register 6, offset: 0x103C */
    __IO uint32_t SCGC7;           /**< System Clock Gating Control Register 7, offset: 0x1040 */
    __IO uint32_t CLKDIV1;         /**< System Clock Divider Register 1, offset: 0x1044 */
    uint8_t RESERVED_5[4];
    __IO uint32_t FCFG1;           /**< Flash Configuration Register 1, offset: 0x104C */
    __I uint32_t FCFG2;           /**< Flash Configuration Register 2, offset: 0x1050 */
    uint8_t RESERVED_6[4];
    __I uint32_t UIDMH;            /**< Unique Identification Register Mid-High, offset: 0x1058 */
    __I uint32_t UIDML;            /**< Unique Identification Register Mid Low, offset: 0x105C */
    __I uint32_t UIDL;             /**< Unique Identification Register Low, offset: 0x1060 */
    uint8_t RESERVED_7[156];
    __IO uint32_t COPC;            /**< COP Control Register, offset: 0x1100 */
    __O uint32_t SRVCOP;           /**< Service COP Register, offset: 0x1104 */

} SIM_Type, *SIM_MemMapPtr;

```

```

/*
-----  

-- SIM - Register accessor macros  

----- */

```

```

/*!

* @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros

* @{

*/



/* SIM - Register accessors */

#define SIM_SOPT1_REG(base)          ((base)->SOPT1)
#define SIM_SOPT1CFG_REG(base)       ((base)->SOPT1CFG)
#define SIM_SOPT2_REG(base)          ((base)->SOPT2)
#define SIM_SOPT4_REG(base)          ((base)->SOPT4)
#define SIM_SOPT5_REG(base)          ((base)->SOPT5)
#define SIM_SOPT7_REG(base)          ((base)->SOPT7)
#define SIM_SDID_REG(base)           ((base)->SDID)
#define SIM_SCGC4_REG(base)          ((base)->SCGC4)
#define SIM_SCGC5_REG(base)          ((base)->SCGC5)
#define SIM_SCGC6_REG(base)          ((base)->SCGC6)
#define SIM_SCGC7_REG(base)          ((base)->SCGC7)
#define SIM_CLKDIV1_REG(base)         ((base)->CLKDIV1)
#define SIM_FCFG1_REG(base)          ((base)->FCFG1)
#define SIM_FCFG2_REG(base)          ((base)->FCFG2)
#define SIM_UIDMH_REG(base)          ((base)->UIDMH)
#define SIM_UIDML_REG(base)          ((base)->UIDML)
#define SIM_UIDL_REG(base)           ((base)->UIDL)
#define SIM_COPC_REG(base)           ((base)->COPC)
#define SIM_SRVCOP_REG(base)         ((base)->SRVCOP)

!



* @}

```

```
/* /* end of group SIM_Register_Accessor_Macros */
```

```
/* -----  
-- SIM Register Masks  
----- */
```

```
/*!  
 * @addtogroup SIM_Register_Masks SIM Register Masks  
 * @{  
 */
```

```
/* SOPT1 Bit Fields */  
  
#define SIM_SOPT1_OSC32KSEL_MASK      0xC0000u  
#define SIM_SOPT1_OSC32KSEL_SHIFT     18  
#define SIM_SOPT1_OSC32KSEL_WIDTH     2  
  
#define SIM_SOPT1_OSC32KSEL(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_OSC32KSEL_SHIFT))&SIM_SOPT1_OSC32KSEL_MASK)  
  
#define SIM_SOPT1_USBVSTBY_MASK       0x20000000u  
#define SIM_SOPT1_USBVSTBY_SHIFT     29  
#define SIM_SOPT1_USBVSTBY_WIDTH     1  
  
#define SIM_SOPT1_USBVSTBY(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBVSTBY_SHIFT))&SIM_SOPT1_USBVSTBY_MASK)  
  
#define SIM_SOPT1_USBSSTBY_MASK      0x40000000u  
#define SIM_SOPT1_USBSSTBY_SHIFT     30  
#define SIM_SOPT1_USBSSTBY_WIDTH     1  
  
#define SIM_SOPT1_USBSSTBY(x)         (((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBSSTBY_SHIFT))&SIM_SOPT1_USBSSTBY_MASK)  
  
#define SIM_SOPT1_USBREGEN_MASK      0x80000000u  
#define SIM_SOPT1_USBREGEN_SHIFT     31
```

```

#define SIM_SOPT1_USBREGEN_WIDTH          1

#define SIM_SOPT1_USBREGEN(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1_USBREGEN_SHIFT))&SIM_SOPT1_USBREGEN_MASK)

/* SOPT1CFG Bit Fields */

#define SIM_SOPT1CFG_URWE_MASK           0x1000000u
#define SIM_SOPT1CFG_URWE_SHIFT         24
#define SIM_SOPT1CFG_URWE_WIDTH         1

#define SIM_SOPT1CFG_URWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_URWE_SHIFT))&SIM_SOPT1CFG_URWE_MASK)

#define SIM_SOPT1CFG_UVSWE_MASK          0x2000000u
#define SIM_SOPT1CFG_UVSWE_SHIFT        25
#define SIM_SOPT1CFG_UVSWE_WIDTH         1

#define SIM_SOPT1CFG_UVSWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_UVSWE_SHIFT))&SIM_SOPT1CFG_UVSWE_MASK)

#define SIM_SOPT1CFG_USSWE_MASK          0x4000000u
#define SIM_SOPT1CFG_USSWE_SHIFT        26
#define SIM_SOPT1CFG_USSWE_WIDTH         1

#define SIM_SOPT1CFG_USSWE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT1CFG_USSWE_SHIFT))&SIM_SOPT1CFG_USSWE_MASK)

/* SOPT2 Bit Fields */

#define SIM_SOPT2_RTCCLKOUTSEL_MASK      0x10u
#define SIM_SOPT2_RTCCLKOUTSEL_SHIFT    4
#define SIM_SOPT2_RTCCLKOUTSEL_WIDTH    1

#define SIM_SOPT2_RTCCLKOUTSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT2_RTCCLKOUTSEL_SHIFT))&SIM_SOPT2_RTCCLKOUTSEL_MASK)

#define SIM_SOPT2_CLKOUTSEL_MASK         0xE0u
#define SIM_SOPT2_CLKOUTSEL_SHIFT       5
#define SIM_SOPT2_CLKOUTSEL_WIDTH       3

#define SIM_SOPT2_CLKOUTSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT2_CLKOUTSEL_SHIFT))&SIM_SOPT2_CLKOUTSEL_MASK)

#define SIM_SOPT2_PLLFLLSEL_MASK        0x10000u

```

```

#define SIM_SOPT2_PLLFLLSEL_SHIFT      16
#define SIM_SOPT2_PLLFLLSEL_WIDTH      1
#define SIM_SOPT2_PLLFLLSEL(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_PLLFLLSEL_SHIFT))&SIM_SOPT2_PLLFLLSEL_MASK)
#define SIM_SOPT2_USBSRC_MASK          0x40000u
#define SIM_SOPT2_USBSRC_SHIFT         18
#define SIM_SOPT2_USBSRC_WIDTH         1
#define SIM_SOPT2_USBSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_USBSRC_SHIFT))&SIM_SOPT2_USBSRC_MASK)
#define SIM_SOPT2_TPMSRC_MASK          0x3000000u
#define SIM_SOPT2_TPMSRC_SHIFT         24
#define SIM_SOPT2_TPMSRC_WIDTH         2
#define SIM_SOPT2_TPMSRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_TPMSRC_SHIFT))&SIM_SOPT2_TPMSRC_MASK)
#define SIM_SOPT2_UART0SRC_MASK         0xC000000u
#define SIM_SOPT2_UART0SRC_SHIFT        26
#define SIM_SOPT2_UART0SRC_WIDTH        2
#define SIM_SOPT2_UART0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT2_UART0SRC_SHIFT))&SIM_SOPT2_UART0SRC_MASK)

/* SOPT4 Bit Fields */

#define SIM_SOPT4 TPM1CH0SRC_MASK       0x40000u
#define SIM_SOPT4 TPM1CH0SRC_SHIFT      18
#define SIM_SOPT4 TPM1CH0SRC_WIDTH      1
#define SIM_SOPT4 TPM1CH0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM1CH0SRC_SHIFT))&SIM_SOPT4 TPM1CH0SRC_MASK)
#define SIM_SOPT4 TPM2CH0SRC_MASK       0x100000u
#define SIM_SOPT4 TPM2CH0SRC_SHIFT      20
#define SIM_SOPT4 TPM2CH0SRC_WIDTH      1
#define SIM_SOPT4 TPM2CH0SRC(x) (((uint32_t)((uint32_t)(x)<<SIM_SOPT4 TPM2CH0SRC_SHIFT))&SIM_SOPT4 TPM2CH0SRC_MASK)
#define SIM_SOPT4 TPM0CLKSEL_MASK       0x1000000u

```

```

#define SIM_SOPT4_TPM0CLKSEL_SHIFT      24
#define SIM_SOPT4_TPM0CLKSEL_WIDTH      1
#define SIM_SOPT4_TPM0CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4_TPM0CLKSEL_SHIFT))&SIM_SOPT4_TPM0CLKSEL_MASK)

#define SIM_SOPT4_TPM1CLKSEL_MASK      0x2000000u
#define SIM_SOPT4_TPM1CLKSEL_SHIFT     25
#define SIM_SOPT4_TPM1CLKSEL_WIDTH     1
#define SIM_SOPT4_TPM1CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4_TPM1CLKSEL_SHIFT))&SIM_SOPT4_TPM1CLKSEL_MASK)

#define SIM_SOPT4_TPM2CLKSEL_MASK      0x4000000u
#define SIM_SOPT4_TPM2CLKSEL_SHIFT     26
#define SIM_SOPT4_TPM2CLKSEL_WIDTH     1
#define SIM_SOPT4_TPM2CLKSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT4_TPM2CLKSEL_SHIFT))&SIM_SOPT4_TPM2CLKSEL_MASK)

/* SOPT5 Bit Fields */

#define SIM_SOPT5_UART0TXSRC_MASK      0x3u
#define SIM_SOPT5_UART0TXSRC_SHIFT     0
#define SIM_SOPT5_UART0TXSRC_WIDTH     2
#define SIM_SOPT5_UART0TXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART0TXSRC_SHIFT))&SIM_SOPT5_UART0TXSRC_MASK)

#define SIM_SOPT5_UART0RXSRC_MASK      0x4u
#define SIM_SOPT5_UART0RXSRC_SHIFT     2
#define SIM_SOPT5_UART0RXSRC_WIDTH     1
#define SIM_SOPT5_UART0RXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART0RXSRC_SHIFT))&SIM_SOPT5_UART0RXSRC_MASK)

#define SIM_SOPT5_UART1TXSRC_MASK      0x30u
#define SIM_SOPT5_UART1TXSRC_SHIFT     4
#define SIM_SOPT5_UART1TXSRC_WIDTH     2
#define SIM_SOPT5_UART1TXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART1TXSRC_SHIFT))&SIM_SOPT5_UART1TXSRC_MASK)

#define SIM_SOPT5_UART1RXSRC_MASK      0x40u

```

```

#define SIM_SOPT5_UART1RXSRC_SHIFT      6
#define SIM_SOPT5_UART1RXSRC_WIDTH      1
#define SIM_SOPT5_UART1RXSRC(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART1RXSRC_SHIFT))&SIM_SOPT5_UART1RXSRC_MASK)

#define SIM_SOPT5_UART0ODE_MASK         0x10000u
#define SIM_SOPT5_UART0ODE_SHIFT        16
#define SIM_SOPT5_UART0ODE_WIDTH        1
#define SIM_SOPT5_UART0ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART0ODE_SHIFT))&SIM_SOPT5_UART0ODE_MASK)

#define SIM_SOPT5_UART1ODE_MASK         0x20000u
#define SIM_SOPT5_UART1ODE_SHIFT        17
#define SIM_SOPT5_UART1ODE_WIDTH        1
#define SIM_SOPT5_UART1ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART1ODE_SHIFT))&SIM_SOPT5_UART1ODE_MASK)

#define SIM_SOPT5_UART2ODE_MASK         0x40000u
#define SIM_SOPT5_UART2ODE_SHIFT        18
#define SIM_SOPT5_UART2ODE_WIDTH        1
#define SIM_SOPT5_UART2ODE(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT5_UART2ODE_SHIFT))&SIM_SOPT5_UART2ODE_MASK)

/* SOPT7 Bit Fields */

#define SIM_SOPT7_ADC0TRGSEL_MASK       0xFu
#define SIM_SOPT7_ADC0TRGSEL_SHIFT      0
#define SIM_SOPT7_ADC0TRGSEL_WIDTH      4
#define SIM_SOPT7_ADC0TRGSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADC0TRGSEL_SHIFT))&SIM_SOPT7_ADC0TRGSEL_MASK)

#define SIM_SOPT7_ADCOPRETRGSEL_MASK    0x10u
#define SIM_SOPT7_ADCOPRETRGSEL_SHIFT   4
#define SIM_SOPT7_ADCOPRETRGSEL_WIDTH   1
#define SIM_SOPT7_ADCOPRETRGSEL(x)
(((uint32_t)((uint32_t)(x)<<SIM_SOPT7_ADCOPRETRGSEL_SHIFT))&SIM_SOPT7_ADCOPRETRGSEL_MASK)

```

```

#define SIM_SOPT7_ADCOALTTRGEN_MASK      0x80u
#define SIM_SOPT7_ADCOALTTRGEN_SHIFT     7
#define SIM_SOPT7_ADCOALTTRGEN_WIDTH     1
#define SIM_SOPT7_ADCOALTTRGEN(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADCOALTTRGEN_SHIFT))&SIM_SOPT7_ADCOALTTRGEN_MASK)
/* SDID Bit Fields */

#define SIM_SDID_PINID_MASK            0xFu
#define SIM_SDID_PINID_SHIFT           0
#define SIM_SDID_PINID_WIDTH          4
#define SIM_SDID_PINID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_PINID_SHIFT))&SIM_SDID_PINID_MASK)

#define SIM_SDID_DIEID_MASK           0xF80u
#define SIM_SDID_DIEID_SHIFT          7
#define SIM_SDID_DIEID_WIDTH          5
#define SIM_SDID_DIEID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_DIEID_SHIFT))&SIM_SDID_DIEID_MASK)

#define SIM_SDID_REVID_MASK           0xF000u
#define SIM_SDID_REVID_SHIFT          12
#define SIM_SDID_REVID_WIDTH          4
#define SIM_SDID_REVID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_REVID_SHIFT))&SIM_SDID_REVID_MASK)

#define SIM_SDID_SRAMSIZE_MASK         0xF0000u
#define SIM_SDID_SRAMSIZE_SHIFT        16
#define SIM_SDID_SRAMSIZE_WIDTH        4
#define SIM_SDID_SRAMSIZE(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_SRAMSIZE_SHIFT))&SIM_SDID_SRAMSIZE_MASK)

#define SIM_SDID_SERIESID_MASK         0xF00000u
#define SIM_SDID_SERIESID_SHIFT        20
#define SIM_SDID_SERIESID_WIDTH        4
#define SIM_SDID_SERIESID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_SERIESID_SHIFT))&SIM_SDID_SERIESID_MASK)

```

```

#define SIM_SDID_SUBFAMID_MASK          0xF000000u
#define SIM_SDID_SUBFAMID_SHIFT         24
#define SIM_SDID_SUBFAMID_WIDTH         4
#define SIM_SDID_SUBFAMID(x)           (((uint32_t)((uint32_t)(x))<<SIM_SDID_SUBFAMID_SHIFT))&SIM_SDID_SUBFAMID_MASK)

#define SIM_SDID_FAMID_MASK            0xF0000000u
#define SIM_SDID_FAMID_SHIFT           28
#define SIM_SDID_FAMID_WIDTH           4
#define SIM_SDID_FAMID(x)             (((uint32_t)((uint32_t)(x))<<SIM_SDID_FAMID_SHIFT))&SIM_SDID_FAMID_MASK)

/* SCGC4 Bit Fields */

#define SIM_SCGC4_I2C0_MASK            0x40u
#define SIM_SCGC4_I2C0_SHIFT           6
#define SIM_SCGC4_I2C0_WIDTH           1
#define SIM_SCGC4_I2C0(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_I2C0_SHIFT))&SIM_SCGC4_I2C0_MASK)

#define SIM_SCGC4_I2C1_MASK            0x80u
#define SIM_SCGC4_I2C1_SHIFT           7
#define SIM_SCGC4_I2C1_WIDTH           1
#define SIM_SCGC4_I2C1(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_I2C1_SHIFT))&SIM_SCGC4_I2C1_MASK)

#define SIM_SCGC4_UART0_MASK           0x400u
#define SIM_SCGC4_UART0_SHIFT          10
#define SIM_SCGC4_UART0_WIDTH          1
#define SIM_SCGC4_UART0(x)             (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_UART0_SHIFT))&SIM_SCGC4_UART0_MASK)

#define SIM_SCGC4_UART1_MASK           0x800u
#define SIM_SCGC4_UART1_SHIFT          11
#define SIM_SCGC4_UART1_WIDTH          1
#define SIM_SCGC4_UART1(x)             (((uint32_t)((uint32_t)(x))<<SIM_SCGC4_UART1_SHIFT))&SIM_SCGC4_UART1_MASK)

```

```

#define SIM_SCGC4_UART2_MASK          0x1000u
#define SIM_SCGC4_UART2_SHIFT         12
#define SIM_SCGC4_UART2_WIDTH         1
#define SIM_SCGC4_UART2(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_UART2_SHIFT))&SIM_SCGC4_UART2_MASK)
#define SIM_SCGC4_USBOTG_MASK         0x40000u
#define SIM_SCGC4_USBOTG_SHIFT        18
#define SIM_SCGC4_USBOTG_WIDTH        1
#define SIM_SCGC4_USBOTG(x)           (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_USBOTG_SHIFT))&SIM_SCGC4_USBOTG_MASK)
#define SIM_SCGC4_CMP_MASK            0x80000u
#define SIM_SCGC4_CMP_SHIFT           19
#define SIM_SCGC4_CMP_WIDTH           1
#define SIM_SCGC4_CMP(x)              (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_CMP_SHIFT))&SIM_SCGC4_CMP_MASK)
#define SIM_SCGC4_SPI0_MASK           0x400000u
#define SIM_SCGC4_SPI0_SHIFT          22
#define SIM_SCGC4_SPI0_WIDTH          1
#define SIM_SCGC4_SPI0(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI0_SHIFT))&SIM_SCGC4_SPI0_MASK)
#define SIM_SCGC4_SPI1_MASK           0x800000u
#define SIM_SCGC4_SPI1_SHIFT          23
#define SIM_SCGC4_SPI1_WIDTH          1
#define SIM_SCGC4_SPI1(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC4_SPI1_SHIFT))&SIM_SCGC4_SPI1_MASK)
/* SCGC5 Bit Fields */
#define SIM_SCGC5_LPTMR_MASK          0x1u
#define SIM_SCGC5_LPTMR_SHIFT         0
#define SIM_SCGC5_LPTMR_WIDTH         1
#define SIM_SCGC5_LPTMR(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_LPTMR_SHIFT))&SIM_SCGC5_LPTMR_MASK)

```

```

#define SIM_SCGC5_TSI_MASK          0x20u
#define SIM_SCGC5_TSI_SHIFT         5
#define SIM_SCGC5_TSI_WIDTH         1
#define SIM_SCGC5_TSI(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_TSI_SHIFT))&SIM_SCGC5_TSI_MASK)
#define SIM_SCGC5_PORTA_MASK        0x200u
#define SIM_SCGC5_PORTA_SHIFT       9
#define SIM_SCGC5_PORTA_WIDTH       1
#define SIM_SCGC5_PORTA(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTA_SHIFT))&SIM_SCGC5_PORTA_MASK)
#define SIM_SCGC5_PORTB_MASK        0x400u
#define SIM_SCGC5_PORTB_SHIFT       10
#define SIM_SCGC5_PORTB_WIDTH       1
#define SIM_SCGC5_PORTB(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTB_SHIFT))&SIM_SCGC5_PORTB_MASK)
#define SIM_SCGC5_PORTC_MASK        0x800u
#define SIM_SCGC5_PORTC_SHIFT       11
#define SIM_SCGC5_PORTC_WIDTH       1
#define SIM_SCGC5_PORTC(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTC_SHIFT))&SIM_SCGC5_PORTC_MASK)
#define SIM_SCGC5_PORTD_MASK        0x1000u
#define SIM_SCGC5_PORTD_SHIFT       12
#define SIM_SCGC5_PORTD_WIDTH       1
#define SIM_SCGC5_PORTD(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5_PORTD_SHIFT))&SIM_SCGC5_PORTD_MASK)
#define SIM_SCGC5 PORTE_MASK        0x2000u
#define SIM_SCGC5 PORTE_SHIFT       13
#define SIM_SCGC5 PORTE_WIDTH       1
#define SIM_SCGC5 PORTE(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC5 PORTE_SHIFT))&SIM_SCGC5 PORTE_MASK)
/* SCGC6 Bit Fields */

```

```

#define SIM_SCGC6_FTF_MASK          0x1u
#define SIM_SCGC6_FTF_SHIFT         0
#define SIM_SCGC6_FTF_WIDTH         1
#define SIM_SCGC6_FTF(x)            (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_FTF_SHIFT))&SIM_SCGC6_FTF_MASK)

#define SIM_SCGC6_DMAMUX_MASK        0x2u
#define SIM_SCGC6_DMAMUX_SHIFT       1
#define SIM_SCGC6_DMAMUX_WIDTH       1
#define SIM_SCGC6_DMAMUX(x)          (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_DMAMUX_SHIFT))&SIM_SCGC6_DMAMUX_MASK)

#define SIM_SCGC6_PIT_MASK           0x800000u
#define SIM_SCGC6_PIT_SHIFT          23
#define SIM_SCGC6_PIT_WIDTH          1
#define SIM_SCGC6_PIT(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_PIT_SHIFT))&SIM_SCGC6_PIT_MASK)

#define SIM_SCGC6 TPM0_MASK           0x1000000u
#define SIM_SCGC6 TPM0_SHIFT          24
#define SIM_SCGC6 TPM0_WIDTH          1
#define SIM_SCGC6 TPM0(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM0_SHIFT))&SIM_SCGC6 TPM0_MASK)

#define SIM_SCGC6 TPM1_MASK           0x2000000u
#define SIM_SCGC6 TPM1_SHIFT          25
#define SIM_SCGC6 TPM1_WIDTH          1
#define SIM_SCGC6 TPM1(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM1_SHIFT))&SIM_SCGC6 TPM1_MASK)

#define SIM_SCGC6 TPM2_MASK           0x4000000u
#define SIM_SCGC6 TPM2_SHIFT          26
#define SIM_SCGC6 TPM2_WIDTH          1
#define SIM_SCGC6 TPM2(x)             (((uint32_t)((uint32_t)(x)<<SIM_SCGC6 TPM2_SHIFT))&SIM_SCGC6 TPM2_MASK)

#define SIM_SCGC6 ADC0_MASK           0x8000000u

```

```

#define SIM_SCGC6_ADC0_SHIFT          27
#define SIM_SCGC6_ADC0_WIDTH          1
#define SIM_SCGC6_ADC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_ADC0_SHIFT))&SIM_SCGC6_ADC0_MASK)
#define SIM_SCGC6_RTC_MASK            0x20000000u
#define SIM_SCGC6_RTC_SHIFT           29
#define SIM_SCGC6_RTC_WIDTH           1
#define SIM_SCGC6_RTC(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_RTC_SHIFT))&SIM_SCGC6_RTC_MASK)
#define SIM_SCGC6_DAC0_MASK            0x80000000u
#define SIM_SCGC6_DAC0_SHIFT           31
#define SIM_SCGC6_DAC0_WIDTH           1
#define SIM_SCGC6_DAC0(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC6_DAC0_SHIFT))&SIM_SCGC6_DAC0_MASK)

/* SCGC7 Bit Fields */
#define SIM_SCGC7_DMA_MASK             0x100u
#define SIM_SCGC7_DMA_SHIFT             8
#define SIM_SCGC7_DMA_WIDTH             1
#define SIM_SCGC7_DMA(x) (((uint32_t)((uint32_t)(x)<<SIM_SCGC7_DMA_SHIFT))&SIM_SCGC7_DMA_MASK)

/* CLKDIV1 Bit Fields */
#define SIM_CLKDIV1_OUTDIV4_MASK        0x70000u
#define SIM_CLKDIV1_OUTDIV4_SHIFT       16
#define SIM_CLKDIV1_OUTDIV4_WIDTH       3
#define SIM_CLKDIV1_OUTDIV4(x) (((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV4_SHIFT))&SIM_CLKDIV1_OUTDIV4_MASK)
#define SIM_CLKDIV1_OUTDIV1_MASK        0xF0000000u
#define SIM_CLKDIV1_OUTDIV1_SHIFT       28
#define SIM_CLKDIV1_OUTDIV1_WIDTH       4
#define SIM_CLKDIV1_OUTDIV1(x) (((uint32_t)((uint32_t)(x)<<SIM_CLKDIV1_OUTDIV1_SHIFT))&SIM_CLKDIV1_OUTDIV1_MASK)

```

```

/* FCFG1 Bit Fields */

#define SIM_FCFG1_FLASHDIS_MASK          0x1u
#define SIM_FCFG1_FLASHDIS_SHIFT        0
#define SIM_FCFG1_FLASHDIS_WIDTH        1
#define SIM_FCFG1_FLASHDIS(x)           (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDIS_SHIFT))&SIM_FCFG1_FLASHDIS_MASK)

#define SIM_FCFG1_FLASHDOZE_MASK         0x2u
#define SIM_FCFG1_FLASHDOZE_SHIFT       1
#define SIM_FCFG1_FLASHDOZE_WIDTH       1
#define SIM_FCFG1_FLASHDOZE(x)          (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDOZE_SHIFT))&SIM_FCFG1_FLASHDOZE_MASK)

#define SIM_FCFG1_PFSIZE_MASK           0xF000000u
#define SIM_FCFG1_PFSIZE_SHIFT          24
#define SIM_FCFG1_PFSIZE_WIDTH          4
#define SIM_FCFG1_PFSIZE(x)             (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_PFSIZE_SHIFT))&SIM_FCFG1_PFSIZE_MASK)

/* FCFG2 Bit Fields */

#define SIM_FCFG2_MAXADDR0_MASK          0x7F000000u
#define SIM_FCFG2_MAXADDR0_SHIFT         24
#define SIM_FCFG2_MAXADDR0_WIDTH         7
#define SIM_FCFG2_MAXADDR0(x)            (((uint32_t)((uint32_t)(x))<<SIM_FCFG2_MAXADDR0_SHIFT))&SIM_FCFG2_MAXADDR0_MASK)

/* UIDMH Bit Fields */

#define SIM_UIDMH_UID_MASK              0xFFFFu
#define SIM_UIDMH_UID_SHIFT             0
#define SIM_UIDMH_UID_WIDTH             16
#define SIM_UIDMH_UID(x)                (((uint32_t)((uint32_t)(x))<<SIM_UIDMH_UID_SHIFT))&SIM_UIDMH_UID_MASK)

/* UIDML Bit Fields */

#define SIM_UIDML_UID_MASK              0xFFFFFFFFu

```

```

#define SIM_UIDML_UID_SHIFT          0
#define SIM_UIDML_UID_WIDTH         32
#define SIM_UIDML_UID(x) (((uint32_t)((uint32_t)(x)<<SIM_UIDML_UID_SHIFT))&SIM_UIDML_UID_MASK)

/* UIDL Bit Fields */

#define SIM_UIDL_UID_MASK           0xFFFFFFFFu
#define SIM_UIDL_UID_SHIFT          0
#define SIM_UIDL_UID_WIDTH         32
#define SIM_UIDL_UID(x) (((uint32_t)((uint32_t)(x)<<SIM_UIDL_UID_SHIFT))&SIM_UIDL_UID_MASK)

/* COPC Bit Fields */

#define SIM_COPC_COPW_MASK          0x1u
#define SIM_COPC_COPW_SHIFT          0
#define SIM_COPC_COPW_WIDTH         1
#define SIM_COPC_COPW(x) (((uint32_t)((uint32_t)(x)<<SIM_COPC_COPW_SHIFT))&SIM_COPC_COPW_MASK)

#define SIM_COPC_COPCLKS_MASK        0x2u
#define SIM_COPC_COPCLKS_SHIFT       1
#define SIM_COPC_COPCLKS_WIDTH       1
#define SIM_COPC_COPCLKS(x) (((uint32_t)((uint32_t)(x)<<SIM_COPC_COPCLKS_SHIFT))&SIM_COPC_COPCLKS_MASK)

#define SIM_COPC_COPT_MASK           0xCu
#define SIM_COPC_COPT_SHIFT          2
#define SIM_COPC_COPT_WIDTH          2
#define SIM_COPC_COPT(x) (((uint32_t)((uint32_t)(x)<<SIM_COPC_COPT_SHIFT))&SIM_COPC_COPT_MASK)

/* SRVCOP Bit Fields */

#define SIM_SRVCOP_SRVCOP_MASK        0xFFu
#define SIM_SRVCOP_SRVCOP_SHIFT       0
#define SIM_SRVCOP_SRVCOP_WIDTH       8

```

```

#define SIM_SRVCOP_SRVCOP(x)
(((uint32_t)((uint32_t)(x)<<SIM_SRVCOP_SRVCOP_SHIFT))&SIM_SRVCOP_SRVCOP_MASK)

/*
 * @}
 */

/* SIM - Peripheral instance base addresses */

/** Peripheral SIM base address */

#define SIM_BASE           (0x40047000u)

/** Peripheral SIM base pointer */

#define SIM             ((SIM_Type *)SIM_BASE)

#define SIM_BASE_PTR      (SIM)

/** Array initializer of SIM peripheral base addresses */

#define SIM_BASE_ADDRS    { SIM_BASE }

/** Array initializer of SIM peripheral base pointers */

#define SIM_BASE_PTRS     { SIM }

/*
-----  

-- SIM - Register accessor macros  

----- */

```

```

/*
 * @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor macros
 * @{
 */


```

```

/* SIM - Register instance definitions */

/* SIM */

#define SIM_SOPT1           SIM_SOPT1_REG(SIM)
#define SIM_SOPT1CFG        SIM_SOPT1CFG_REG(SIM)
#define SIM_SOPT2           SIM_SOPT2_REG(SIM)
#define SIM_SOPT4           SIM_SOPT4_REG(SIM)
#define SIM_SOPT5           SIM_SOPT5_REG(SIM)
#define SIM_SOPT7           SIM_SOPT7_REG(SIM)
#define SIM_SDID            SIM_SDID_REG(SIM)
#define SIM_SCGC4           SIM_SCGC4_REG(SIM)
#define SIM_SCGC5           SIM_SCGC5_REG(SIM)
#define SIM_SCGC6           SIM_SCGC6_REG(SIM)
#define SIM_SCGC7           SIM_SCGC7_REG(SIM)
#define SIM_CLKDIV1         SIM_CLKDIV1_REG(SIM)
#define SIM_FCFG1            SIM_FCFG1_REG(SIM)
#define SIM_FCFG2            SIM_FCFG2_REG(SIM)
#define SIM_UIDMH           SIM_UIDMH_REG(SIM)
#define SIM_UIDML           SIM_UIDML_REG(SIM)
#define SIM_UIDL             SIM_UIDL_REG(SIM)
#define SIM_COPC             SIM_COPC_REG(SIM)
#define SIM_SRVCOP           SIM_SRVCOP_REG(SIM)

/*
 * @}
 */
/* /* end of group SIM_Register_Accessor_Macros */

```

```

/*
 * @}

```

```
*/ /* end of group SIM_Peripheral_Access_Layer */
```

```
/* -----  
-- SMC Peripheral Access Layer  
----- */
```

```
/*!  
* @addtogroup SMC_Peripheral_Access_Layer SMC Peripheral Access Layer  
* @{  
*/
```

```
/** SMC - Register Layout Typedef */  
  
typedef struct {  
  
    __IO uint8_t PMPROT;           /**< Power Mode Protection register, offset: 0x0 */  
    __IO uint8_t PMCTRL;          /**< Power Mode Control register, offset: 0x1 */  
    __IO uint8_t STOPCTRL;         /**< Stop Control Register, offset: 0x2 */  
    __I uint8_t PMSTAT;           /**< Power Mode Status register, offset: 0x3 */  
  
} SMC_Type, *SMC_MemMapPtr;
```

```
/* -----  
-- SMC - Register accessor macros  
----- */
```

```
/*!  
* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros  
* @{  
*/
```

```

/* SMC - Register accessors */

#define SMC_PMPROT_REG(base)          ((base)->PMPROT)
#define SMC_PMCTRL_REG(base)          ((base)->PMCTRL)
#define SMC_STOPCTRL_REG(base)        ((base)->STOPCTRL)
#define SMC_PMSTAT_REG(base)          ((base)->PMSTAT)

/*!
 * @}
 */

/* /* end of group SMC_Register_Accessor_Macros */

/*
----- --
-- SMC Register Masks
----- */
/*!

 * @addtogroup SMC_Register_Masks SMC Register Masks
 * @{
 */

/* PMPROT Bit Fields */

#define SMC_PMPROT_AVLLS_MASK         0x2u
#define SMC_PMPROT_AVLLS_SHIFT        1
#define SMC_PMPROT_AVLLS_WIDTH        1

#define SMC_PMPROT_AVLLS(x)           (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLLS_SHIFT))&SMC_PMPROT_AVLLS_MASK)

#define SMC_PMPROT_ALLS_MASK          0x8u
#define SMC_PMPROT_ALLS_SHIFT         3

```

```

#define SMC_PMPROT_ALLS_WIDTH          1
#define SMC_PMPROT_ALLS(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_ALLS_SHIFT))&SMC_PMPROT_ALLS_MASK)
#define SMC_PMPROT_AVLP_MASK          0x20u
#define SMC_PMPROT_AVLP_SHIFT         5
#define SMC_PMPROT_AVLP_WIDTH         1
#define SMC_PMPROT_AVLP(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMPROT_AVLP_SHIFT))&SMC_PMPROT_AVLP_MASK)

/* PMCTRL Bit Fields */

#define SMC_PMCTRL_STOPM_MASK          0x7u
#define SMC_PMCTRL_STOPM_SHIFT         0
#define SMC_PMCTRL_STOPM_WIDTH         3
#define SMC_PMCTRL_STOPM(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPM_SHIFT))&SMC_PMCTRL_STOPM_MASK)
#define SMC_PMCTRL_STOPA_MASK          0x8u
#define SMC_PMCTRL_STOPA_SHIFT         3
#define SMC_PMCTRL_STOPA_WIDTH         1
#define SMC_PMCTRL_STOPA(x)            (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_STOPA_SHIFT))&SMC_PMCTRL_STOPA_MASK)
#define SMC_PMCTRL_RUNM_MASK           0x60u
#define SMC_PMCTRL_RUNM_SHIFT          5
#define SMC_PMCTRL_RUNM_WIDTH          2
#define SMC_PMCTRL_RUNM(x)             (((uint8_t)((uint8_t)(x)<<SMC_PMCTRL_RUNM_SHIFT))&SMC_PMCTRL_RUNM_MASK)

/* STOPCTRL Bit Fields */

#define SMC_STOPCTRL_VLLSM_MASK        0x7u
#define SMC_STOPCTRL_VLLSM_SHIFT       0
#define SMC_STOPCTRL_VLLSM_WIDTH       3
#define SMC_STOPCTRL_VLLSM(x)          (((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_VLLSM_SHIFT))&SMC_STOPCTRL_VLLSM_MASK)
#define SMC_STOPCTRL_PORPO_MASK        0x20u

```

```

#define SMC_STOPCTRL_PORPO_SHIFT      5
#define SMC_STOPCTRL_PORPO_WIDTH     1
#define SMC_STOPCTRL_PORPO(x)
(((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PORPO_SHIFT))&SMC_STOPCTRL_PORPO_MASK)

#define SMC_STOPCTRL_PSTOPO_MASK      0xC0u
#define SMC_STOPCTRL_PSTOPO_SHIFT     6
#define SMC_STOPCTRL_PSTOPO_WIDTH     2
#define SMC_STOPCTRL_PSTOPO(x)
(((uint8_t)((uint8_t)(x)<<SMC_STOPCTRL_PSTOPO_SHIFT))&SMC_STOPCTRL_PSTOPO_MASK)

/* PMSTAT Bit Fields */

#define SMC_PMSTAT_PMSTAT_MASK      0x7Fu
#define SMC_PMSTAT_PMSTAT_SHIFT     0
#define SMC_PMSTAT_PMSTAT_WIDTH     7
#define SMC_PMSTAT_PMSTAT(x)
(((uint8_t)((uint8_t)(x)<<SMC_PMSTAT_PMSTAT_SHIFT))&SMC_PMSTAT_PMSTAT_MASK)

/*
 */
/* @}
 */
/* end of group SMC_Register_Masks */

```

```

/* SMC - Peripheral instance base addresses */
/** Peripheral SMC base address */
#define SMC_BASE          (0x4007E000u)
/** Peripheral SMC base pointer */
#define SMC          ((SMC_Type *)SMC_BASE)
#define SMC_BASE_PTR      (SMC)
/** Array initializer of SMC peripheral base addresses */
#define SMC_BASE_ADDRS    { SMC_BASE }
/** Array initializer of SMC peripheral base pointers */

```

```

#define SMC_BASE_PTRS          { SMC }

/*
-----  

-- SMC - Register accessor macros  

----- */
/*!  

* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor macros  

* @{
* /  

  

/* SMC - Register instance definitions */  

/* SMC */  

#define SMC_PMPROT           SMC_PMPROT_REG(SMC)  

#define SMC_PMCTRL            SMC_PMCTRL_REG(SMC)  

#define SMC_STOPCTRL          SMC_STOPCTRL_REG(SMC)  

#define SMC_PMSTAT             SMC_PMSTAT_REG(SMC)  

  

/*!  

* @}  

* /* end of group SMC_Register_Accessor_Macros */  

  

/*!  

* @}  

* /* end of group SMC_Peripheral_Access_Layer */

```

```

/*
-- SPI Peripheral Access Layer
*/
/*!
* @addtogroup SPI_Peripheral_Access_Layer SPI Peripheral Access Layer
* @{
*/
/** SPI - Register Layout Typedef */
typedef struct {
    __IO uint8_t C1;           /**< SPI control register 1, offset: 0x0 */
    __IO uint8_t C2;           /**< SPI control register 2, offset: 0x1 */
    __IO uint8_t BR;           /**< SPI baud rate register, offset: 0x2 */
    __IO uint8_t S;            /**< SPI status register, offset: 0x3 */
    uint8_t RESERVED_0[1];
    __IO uint8_t D;            /**< SPI data register, offset: 0x5 */
    uint8_t RESERVED_1[1];
    __IO uint8_t M;            /**< SPI match register, offset: 0x7 */
} SPI_Type, *SPI_MemMapPtr;

```

```

/*
-- SPI - Register accessor macros
*/

```

```

/*
* @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros
* @{
*/

```

```
/* SPI - Register accessors */

#define SPI_C1_REG(base)          ((base)->C1)
#define SPI_C2_REG(base)          ((base)->C2)
#define SPI_BR_REG(base)          ((base)->BR)
#define SPI_S_REG(base)           ((base)->S)
#define SPI_D_REG(base)           ((base)->D)
#define SPI_M_REG(base)           ((base)->M)

/*!
 * @}
 */
/* /* end of group SPI_Register_Accessor_Macros */
```

```
/*
-----  
-- SPI Register Masks  
----- */
```

```
/*!  
 * @addtogroup SPI_Register_Masks SPI Register Masks  
 * @{  
 */
```

```
/* C1 Bit Fields */

#define SPI_C1_LSBFE_MASK        0x1u
#define SPI_C1_LSBFE_SHIFT       0
#define SPI_C1_LSBFE_WIDTH       1
```

```

#define SPI_C1_LSBFE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_LSBFE_SHIFT))&SPI_C1_LSBFE_MASK)

#define SPI_C1_SSOE_MASK          0x2u
#define SPI_C1_SSOE_SHIFT         1
#define SPI_C1_SSOE_WIDTH         1

#define SPI_C1_SSOE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SSOE_SHIFT))&SPI_C1_SSOE_MASK)

#define SPI_C1_CPHA_MASK          0x4u
#define SPI_C1_CPHA_SHIFT         2
#define SPI_C1_CPHA_WIDTH         1

#define SPI_C1_CPHA(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_CPHA_SHIFT))&SPI_C1_CPHA_MASK)

#define SPI_C1_CPOL_MASK          0x8u
#define SPI_C1_CPOL_SHIFT         3
#define SPI_C1_CPOL_WIDTH         1

#define SPI_C1_CPOL(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_CPOL_SHIFT))&SPI_C1_CPOL_MASK)

#define SPI_C1_MSTR_MASK          0x10u
#define SPI_C1_MSTR_SHIFT         4
#define SPI_C1_MSTR_WIDTH         1

#define SPI_C1_MSTR(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_MSTR_SHIFT))&SPI_C1_MSTR_MASK)

#define SPI_C1_SPTIE_MASK          0x20u
#define SPI_C1_SPTIE_SHIFT         5
#define SPI_C1_SPTIE_WIDTH         1

#define SPI_C1_SPTIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPTIE_SHIFT))&SPI_C1_SPTIE_MASK)

#define SPI_C1_SPE_MASK           0x40u
#define SPI_C1_SPE_SHIFT          6
#define SPI_C1_SPE_WIDTH          1

```

```

#define SPI_C1_SPE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPE_SHIFT))&SPI_C1_SPE_MASK)

#define SPI_C1_SPIE_MASK          0x80u
#define SPI_C1_SPIE_SHIFT         7
#define SPI_C1_SPIE_WIDTH         1

#define SPI_C1_SPIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C1_SPIE_SHIFT))&SPI_C1_SPIE_MASK)

/* C2 Bit Fields */

#define SPI_C2_SPC0_MASK          0x1u
#define SPI_C2_SPC0_SHIFT         0
#define SPI_C2_SPC0_WIDTH         1

#define SPI_C2_SPC0(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPC0_SHIFT))&SPI_C2_SPC0_MASK)

#define SPI_C2_SPISWAI_MASK        0x2u
#define SPI_C2_SPISWAI_SHIFT       1
#define SPI_C2_SPISWAI_WIDTH       1

#define SPI_C2_SPISWAI(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPISWAI_SHIFT))&SPI_C2_SPISWAI_MASK)

#define SPI_C2_RXDMAE_MASK         0x4u
#define SPI_C2_RXDMAE_SHIFT        2
#define SPI_C2_RXDMAE_WIDTH        1

#define SPI_C2_RXDMAE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_RXDMAE_SHIFT))&SPI_C2_RXDMAE_MASK)

#define SPI_C2_BIDIROE_MASK        0x8u
#define SPI_C2_BIDIROE_SHIFT       3
#define SPI_C2_BIDIROE_WIDTH       1

#define SPI_C2_BIDIROE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_BIDIROE_SHIFT))&SPI_C2_BIDIROE_MASK)

#define SPI_C2_MODFEN_MASK         0x10u
#define SPI_C2_MODFEN_SHIFT        4
#define SPI_C2_MODFEN_WIDTH        1

```

```

#define SPI_C2_MODFEN(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_MODFEN_SHIFT))&SPI_C2_MODFEN_MASK)

#define SPI_C2_TXDMAE_MASK          0x20u
#define SPI_C2_TXDMAE_SHIFT         5
#define SPI_C2_TXDMAE_WIDTH         1

#define SPI_C2_TXDMAE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_TXDMAE_SHIFT))&SPI_C2_TXDMAE_MASK)

#define SPI_C2_SPMIE_MASK           0x80u
#define SPI_C2_SPMIE_SHIFT          7
#define SPI_C2_SPMIE_WIDTH          1

#define SPI_C2_SPMIE(x)
(((uint8_t)((uint8_t)(x)<<SPI_C2_SPMIE_SHIFT))&SPI_C2_SPMIE_MASK)

/* BR Bit Fields */

#define SPI_BR_SPR_MASK             0xFu
#define SPI_BR_SPR_SHIFT            0
#define SPI_BR_SPR_WIDTH            4

#define SPI_BR_SPR(x)
(((uint8_t)((uint8_t)(x)<<SPI_BR_SPR_SHIFT))&SPI_BR_SPR_MASK)

#define SPI_BR_SPPR_MASK             0x70u
#define SPI_BR_SPPR_SHIFT            4
#define SPI_BR_SPPR_WIDTH            3

#define SPI_BR_SPPR(x)
(((uint8_t)((uint8_t)(x)<<SPI_BR_SPPR_SHIFT))&SPI_BR_SPPR_MASK)

/* S Bit Fields */

#define SPI_S_MODF_MASK              0x10u
#define SPI_S_MODF_SHIFT             4
#define SPI_S_MODF_WIDTH              1

#define SPI_S_MODF(x)
(((uint8_t)((uint8_t)(x)<<SPI_S_MODF_SHIFT))&SPI_S_MODF_MASK)

#define SPI_S_SPTEF_MASK             0x20u
#define SPI_S_SPTEF_SHIFT            5

```

```

#define SPI_S_SPTEF_WIDTH          1
#define SPI_S_SPTEF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPTEF_SHIFT))&SPI_S_SPTEF_MASK)
#define SPI_S_SPMF_MASK            0x40u
#define SPI_S_SPMF_SHIFT           6
#define SPI_S_SPMF_WIDTH           1
#define SPI_S_SPMF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPMF_SHIFT))&SPI_S_SPMF_MASK)
#define SPI_S_SPRF_MASK             0x80u
#define SPI_S_SPRF_SHIFT            7
#define SPI_S_SPRF_WIDTH            1
#define SPI_S_SPRF(x) (((uint8_t)((uint8_t)(x)<<SPI_S_SPRF_SHIFT))&SPI_S_SPRF_MASK)

/* D Bit Fields */
#define SPI_D_Bits_MASK             0xFFu
#define SPI_D_Bits_SHIFT             0
#define SPI_D_Bits_WIDTH             8
#define SPI_D_Bits(x) (((uint8_t)((uint8_t)(x)<<SPI_D_Bits_SHIFT))&SPI_D_Bits_MASK)

/* M Bit Fields */
#define SPI_M_Bits_MASK             0xFFu
#define SPI_M_Bits_SHIFT             0
#define SPI_M_Bits_WIDTH             8
#define SPI_M_Bits(x) (((uint8_t)((uint8_t)(x)<<SPI_M_Bits_SHIFT))&SPI_M_Bits_MASK)

```

```

/*
 * @}
 */
 /* end of group SPI_Register_Masks */

```

```

/* SPI - Peripheral instance base addresses */

```

```

/** Peripheral SPI0 base address */
#define SPI0_BASE           (0x40076000u)

/** Peripheral SPI0 base pointer */
#define SPI0          ((SPI_Type *)SPI0_BASE)

#define SPI0_BASE_PTR      (SPI0)

/** Peripheral SPI1 base address */
#define SPI1_BASE           (0x40077000u)

/** Peripheral SPI1 base pointer */
#define SPI1          ((SPI_Type *)SPI1_BASE)

#define SPI1_BASE_PTR      (SPI1)

/** Array initializer of SPI peripheral base addresses */
#define SPI_BASE_ADDRS     { SPI0_BASE, SPI1_BASE }

/** Array initializer of SPI peripheral base pointers */
#define SPI_BASE_PTRS       { SPI0, SPI1 }

/*
-----  

-- SPI - Register accessor macros  

----- */
/*!  

 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor macros  

 * @{
 * /  

  

/* SPI - Register instance definitions */  

/* SPI0 */  

#define SPI0_C1           SPI_C1_REG(SPI0)  

#define SPI0_C2           SPI_C2_REG(SPI0)

```

```

#define SPI0_BR           SPI_BR_REG(SPI0)
#define SPI0_S            SPI_S_REG(SPI0)
#define SPI0_D            SPI_D_REG(SPI0)
#define SPI0_M            SPI_M_REG(SPI0)

/* SPI1 */

#define SPI1_C1           SPI_C1_REG(SPI1)
#define SPI1_C2           SPI_C2_REG(SPI1)
#define SPI1_BR           SPI_BR_REG(SPI1)
#define SPI1_S            SPI_S_REG(SPI1)
#define SPI1_D            SPI_D_REG(SPI1)
#define SPI1_M            SPI_M_REG(SPI1)

/*!
 * @}
 */

/* /* end of group SPI_Register_Accessor_Macros */

/*!
 * @}
 */

/* /* end of group SPI_Peripheral_Access_Layer */

/*
-----*
-- TPM Peripheral Access Layer
----- */
/*!

* @addtogroup TPM_Peripheral_Access_Layer TPM Peripheral Access Layer
* @{

```

```
*/
```

```
/** TPM - Register Layout Typedef */

typedef struct {

    __IO uint32_t SC;           /**< Status and Control, offset: 0x0 */

    __IO uint32_t CNT;          /**< Counter, offset: 0x4 */

    __IO uint32_t MOD;          /**< Modulo, offset: 0x8 */

    struct {                   /* offset: 0xC, array step: 0x8 */

        __IO uint32_t CnSC;      /**< Channel (n) Status and Control, array offset: 0xC, array step: 0x8 */

        __IO uint32_t CnV;       /**< Channel (n) Value, array offset: 0x10, array step: 0x8 */

    } CONTROLS[6];

    uint8_t RESERVED_0[20];

    __IO uint32_t STATUS;       /**< Capture and Compare Status, offset: 0x50 */

    uint8_t RESERVED_1[48];

    __IO uint32_t CONF;         /**< Configuration, offset: 0x84 */

} TPM_Type, *TPM_MemMapPtr;
```

```
/* -----
-- TPM - Register accessor macros
----- */
```

```
/*!
* @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros
* @{
*/

```

```
/* TPM - Register accessors */
```

```
#define TPM_SC_REG(base)          ((base)->SC)
#define TPM_CNT_REG(base)          ((base)->CNT)
#define TPM_MOD_REG(base)          ((base)->MOD)
#define TPM_CnSC_REG(base,index)    ((base)->CONTROLS[index].CnSC)
#define TPM_CnSC_COUNT             6
#define TPM_CnV_REG(base,index)    ((base)->CONTROLS[index].CnV)
#define TPM_CnV_COUNT              6
#define TPM_STATUS_REG(base)        ((base)->STATUS)
#define TPM_CONF_REG(base)          ((base)->CONF)
```

```
/*
 * @}
 */
/* /* end of group TPM_Register_Accessor_Macros */
```

```
/*
-----  
-- TPM Register Masks  
----- */
```

```
/*
 * @addtogroup TPM_Register_Masks TPM Register Masks
 * @{
 */

```

```
/* SC Bit Fields */

#define TPM_SC_PS_MASK           0x7u
#define TPM_SC_PS_SHIFT           0
#define TPM_SC_PS_WIDTH            3
```

```

#define TPM_SC_PS(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_PS_SHIFT))&TPM_SC_PS_MASK)

#define TPM_SC_CMOD_MASK          0x18u
#define TPM_SC_CMOD_SHIFT         3
#define TPM_SC_CMOD_WIDTH         2

#define TPM_SC_CMOD(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_CMOD_SHIFT))&TPM_SC_CMOD_MASK)

#define TPM_SC_CPWMS_MASK         0x20u
#define TPM_SC_CPWMS_SHIFT        5
#define TPM_SC_CPWMS_WIDTH        1

#define TPM_SC_CPWMS(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_CPWMS_SHIFT))&TPM_SC_CPWMS_MASK)

#define TPM_SC_TOIE_MASK          0x40u
#define TPM_SC_TOIE_SHIFT         6
#define TPM_SC_TOIE_WIDTH         1

#define TPM_SC_TOIE(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_TOIE_SHIFT))&TPM_SC_TOIE_MASK)

#define TPM_SC_TOF_MASK           0x80u
#define TPM_SC_TOF_SHIFT          7
#define TPM_SC_TOF_WIDTH          1

#define TPM_SC_TOF(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_TOF_SHIFT))&TPM_SC_TOF_MASK)

#define TPM_SC_DMA_MASK           0x100u
#define TPM_SC_DMA_SHIFT          8
#define TPM_SC_DMA_WIDTH          1

#define TPM_SC_DMA(x)
(((uint32_t)((uint32_t)(x)<<TPM_SC_DMA_SHIFT))&TPM_SC_DMA_MASK)

/* CNT Bit Fields */

#define TPM_CNT_COUNT_MASK        0xFFFFu
#define TPM_CNT_COUNT_SHIFT       0
#define TPM_CNT_COUNT_WIDTH      16

```

```

#define TPM_CNT_COUNT(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CNT_COUNT_SHIFT))&TPM_CNT_COUNT_MASK)

/* MOD Bit Fields */

#define TPM_MOD_MOD_MASK          0xFFFFu
#define TPM_MOD_MOD_SHIFT         0
#define TPM_MOD_MOD_WIDTH         16

#define TPM_MOD_MOD(x)
(((uint32_t) (((uint32_t)(x))<<TPM_MOD_MOD_SHIFT))&TPM_MOD_MOD_MASK)

/* CnSC Bit Fields */

#define TPM_CnSC_DMA_MASK         0x1u
#define TPM_CnSC_DMA_SHIFT        0
#define TPM_CnSC_DMA_WIDTH        1

#define TPM_CnSC_DMA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_DMA_SHIFT))&TPM_CnSC_DMA_MASK)

#define TPM_CnSC_ELSA_MASK         0x4u
#define TPM_CnSC_ELSA_SHIFT        2
#define TPM_CnSC_ELSA_WIDTH        1

#define TPM_CnSC_ELSA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_ELSA_SHIFT))&TPM_CnSC_ELSA_MASK)

#define TPM_CnSC_ELSB_MASK         0x8u
#define TPM_CnSC_ELSB_SHIFT        3
#define TPM_CnSC_ELSB_WIDTH        1

#define TPM_CnSC_ELSB(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_ELSB_SHIFT))&TPM_CnSC_ELSB_MASK)

#define TPM_CnSC_MSA_MASK         0x10u
#define TPM_CnSC_MSA_SHIFT        4
#define TPM_CnSC_MSA_WIDTH        1

#define TPM_CnSC_MSA(x)
(((uint32_t) (((uint32_t)(x))<<TPM_CnSC_MSA_SHIFT))&TPM_CnSC_MSA_MASK)

#define TPM_CnSC_MSB_MASK         0x20u
#define TPM_CnSC_MSB_SHIFT        5

```

```

#define TPM_CnSC_MSB_WIDTH          1
#define TPM_CnSC_MSB(x)             (((uint32_t)((uint32_t)(x)<<TPM_CnSC_MSB_SHIFT))&TPM_CnSC_MSB_MASK)
#define TPM_CnSC_CHIE_MASK          0x40u
#define TPM_CnSC_CHIE_SHIFT         6
#define TPM_CnSC_CHIE_WIDTH         1
#define TPM_CnSC_CHIE(x)            (((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHIE_SHIFT))&TPM_CnSC_CHIE_MASK)
#define TPM_CnSC_CHF_MASK           0x80u
#define TPM_CnSC_CHF_SHIFT          7
#define TPM_CnSC_CHF_WIDTH          1
#define TPM_CnSC_CHF(x)             (((uint32_t)((uint32_t)(x)<<TPM_CnSC_CHF_SHIFT))&TPM_CnSC_CHF_MASK)

/* CnV Bit Fields */

#define TPM_CnV_VAL_MASK            0xFFFFu
#define TPM_CnV_VAL_SHIFT           0
#define TPM_CnV_VAL_WIDTH          16
#define TPM_CnV_VAL(x)              (((uint32_t)((uint32_t)(x)<<TPM_CnV_VAL_SHIFT))&TPM_CnV_VAL_MASK)

/* STATUS Bit Fields */

#define TPM_STATUS_CH0F_MASK         0x1u
#define TPM_STATUS_CH0F_SHIFT        0
#define TPM_STATUS_CH0F_WIDTH        1
#define TPM_STATUS_CH0F(x)           (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH0F_SHIFT))&TPM_STATUS_CH0F_MASK)
#define TPM_STATUS_CH1F_MASK         0x2u
#define TPM_STATUS_CH1F_SHIFT        1
#define TPM_STATUS_CH1F_WIDTH        1
#define TPM_STATUS_CH1F(x)           (((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH1F_SHIFT))&TPM_STATUS_CH1F_MASK)
#define TPM_STATUS_CH2F_MASK         0x4u

```

```

#define TPM_STATUS_CH2F_SHIFT          2
#define TPM_STATUS_CH2F_WIDTH          1
#define TPM_STATUS_CH2F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH2F_SHIFT))&TPM_STATUS_CH2F_MASK)

#define TPM_STATUS_CH3F_MASK           0x8u
#define TPM_STATUS_CH3F_SHIFT          3
#define TPM_STATUS_CH3F_WIDTH          1
#define TPM_STATUS_CH3F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH3F_SHIFT))&TPM_STATUS_CH3F_MASK)

#define TPM_STATUS_CH4F_MASK           0x10u
#define TPM_STATUS_CH4F_SHIFT          4
#define TPM_STATUS_CH4F_WIDTH          1
#define TPM_STATUS_CH4F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH4F_SHIFT))&TPM_STATUS_CH4F_MASK)

#define TPM_STATUS_CH5F_MASK           0x20u
#define TPM_STATUS_CH5F_SHIFT          5
#define TPM_STATUS_CH5F_WIDTH          1
#define TPM_STATUS_CH5F(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_CH5F_SHIFT))&TPM_STATUS_CH5F_MASK)

#define TPM_STATUS_TOF_MASK            0x100u
#define TPM_STATUS_TOF_SHIFT           8
#define TPM_STATUS_TOF_WIDTH           1
#define TPM_STATUS_TOF(x)
(((uint32_t)((uint32_t)(x)<<TPM_STATUS_TOF_SHIFT))&TPM_STATUS_TOF_MASK)

/* CONF Bit Fields */

#define TPM_CONF_DOZEEN_MASK          0x20u
#define TPM_CONF_DOZEEN_SHIFT          5
#define TPM_CONF_DOZEEN_WIDTH          1
#define TPM_CONF_DOZEEN(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_DOZEEN_SHIFT))&TPM_CONF_DOZEEN_MASK)

#define TPM_CONF_DBGMODE_MASK          0xC0u

```

```

#define TPM_CONF_DBGMODE_SHIFT          6
#define TPM_CONF_DBGMODE_WIDTH          2
#define TPM_CONF_DBGMODE(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_DBGMODE_SHIFT))&TPM_CONF_DBGMODE_MASK)

#define TPM_CONF_GTBEEN_MASK           0x200u
#define TPM_CONF_GTBEEN_SHIFT          9
#define TPM_CONF_GTBEEN_WIDTH          1
#define TPM_CONF_GTBEEN(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_GTBEEN_SHIFT))&TPM_CONF_GTBEEN_MASK)

#define TPM_CONF_CSOT_MASK             0x10000u
#define TPM_CONF_CSOT_SHIFT            16
#define TPM_CONF_CSOT_WIDTH            1
#define TPM_CONF_CSOT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOT_SHIFT))&TPM_CONF_CSOT_MASK)

#define TPM_CONF_CSOO_MASK             0x20000u
#define TPM_CONF_CSOO_SHIFT            17
#define TPM_CONF_CSOO_WIDTH            1
#define TPM_CONF_CSOO(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CSOO_SHIFT))&TPM_CONF_CSOO_MASK)

#define TPM_CONF_CROT_MASK             0x40000u
#define TPM_CONF_CROT_SHIFT            18
#define TPM_CONF_CROT_WIDTH            1
#define TPM_CONF_CROT(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_CROT_SHIFT))&TPM_CONF_CROT_MASK)

#define TPM_CONF_TRGSEL_MASK           0xF000000u
#define TPM_CONF_TRGSEL_SHIFT          24
#define TPM_CONF_TRGSEL_WIDTH          4
#define TPM_CONF_TRGSEL(x)
(((uint32_t)((uint32_t)(x)<<TPM_CONF_TRGSEL_SHIFT))&TPM_CONF_TRGSEL_MASK)

```

/*!

```

* @}

/* /* end of group TPM_Register_Masks */

/* TPM - Peripheral instance base addresses */

/** Peripheral TPM0 base address */

#define TPM0_BASE          (0x40038000u)

/** Peripheral TPM0 base pointer */

#define TPM0           ((TPM_Type *)TPM0_BASE)

#define TPM0_BASE_PTR      (TPM0)

/** Peripheral TPM1 base address */

#define TPM1_BASE          (0x40039000u)

/** Peripheral TPM1 base pointer */

#define TPM1           ((TPM_Type *)TPM1_BASE)

#define TPM1_BASE_PTR      (TPM1)

/** Peripheral TPM2 base address */

#define TPM2_BASE          (0x4003A000u)

/** Peripheral TPM2 base pointer */

#define TPM2           ((TPM_Type *)TPM2_BASE)

#define TPM2_BASE_PTR      (TPM2)

/** Array initializer of TPM peripheral base addresses */

#define TPM_BASE_ADDRS     { TPM0_BASE, TPM1_BASE, TPM2_BASE }

/** Array initializer of TPM peripheral base pointers */

#define TPM_BASE_PTRS       { TPM0, TPM1, TPM2 }

/*
----- --
-- TPM - Register accessor macros
----- */

```

```
/*!  
 * @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor macros  
 * @{  
 */
```

```
/* TPM - Register instance definitions */  
/* TPM0 */  
  
#define TPM0_SC TPM_SC_REG(TPM0)  
#define TPM0_CNT TPM_CNT_REG(TPM0)  
#define TPM0_MOD TPM_MOD_REG(TPM0)  
#define TPM0_COSC TPM_CnSC_REG(TPM0,0)  
#define TPM0_COV TPM_CnV_REG(TPM0,0)  
#define TPM0_C1SC TPM_CnSC_REG(TPM0,1)  
#define TPM0_C1V TPM_CnV_REG(TPM0,1)  
#define TPM0_C2SC TPM_CnSC_REG(TPM0,2)  
#define TPM0_C2V TPM_CnV_REG(TPM0,2)  
#define TPM0_C3SC TPM_CnSC_REG(TPM0,3)  
#define TPM0_C3V TPM_CnV_REG(TPM0,3)  
#define TPM0_C4SC TPM_CnSC_REG(TPM0,4)  
#define TPM0_C4V TPM_CnV_REG(TPM0,4)  
#define TPM0_C5SC TPM_CnSC_REG(TPM0,5)  
#define TPM0_C5V TPM_CnV_REG(TPM0,5)  
#define TPM0_STATUS TPM_STATUS_REG(TPM0)  
#define TPM0_CONF TPM_CONF_REG(TPM0)  
  
/* TPM1 */  
  
#define TPM1_SC TPM_SC_REG(TPM1)  
#define TPM1_CNT TPM_CNT_REG(TPM1)  
#define TPM1_MOD TPM_MOD_REG(TPM1)
```

```

#define TPM1_COSC           TPM_CnSC_REG(TPM1,0)
#define TPM1_COV            TPM_CnV_REG(TPM1,0)
#define TPM1_C1SC           TPM_CnSC_REG(TPM1,1)
#define TPM1_C1V            TPM_CnV_REG(TPM1,1)
#define TPM1_STATUS          TPM_STATUS_REG(TPM1)
#define TPM1_CONF            TPM_CONF_REG(TPM1)

/* TPM2 */

#define TPM2_SC              TPM_SC_REG(TPM2)
#define TPM2_CNT             TPM_CNT_REG(TPM2)
#define TPM2_MOD             TPM_MOD_REG(TPM2)
#define TPM2_COSC           TPM_CnSC_REG(TPM2,0)
#define TPM2_COV            TPM_CnV_REG(TPM2,0)
#define TPM2_C1SC           TPM_CnSC_REG(TPM2,1)
#define TPM2_C1V            TPM_CnV_REG(TPM2,1)
#define TPM2_STATUS          TPM_STATUS_REG(TPM2)
#define TPM2_CONF            TPM_CONF_REG(TPM2)

/* TPM - Register array accessors */

#define TPM0_CnSC(index)     TPM_CnSC_REG(TPM0,index)
#define TPM1_CnSC(index)     TPM_CnSC_REG(TPM1,index)
#define TPM2_CnSC(index)     TPM_CnSC_REG(TPM2,index)
#define TPM0_CnV(index)      TPM_CnV_REG(TPM0,index)
#define TPM1_CnV(index)      TPM_CnV_REG(TPM1,index)
#define TPM2_CnV(index)      TPM_CnV_REG(TPM2,index)

/*
 * @@
 */
 /* /* end of group TPM_Register_Accessor_Macros */

```

```
/*!  
 * @}  
 */ /* end of group TPM_Peripheral_Access_Layer */
```

```
/* -----  
-- TSI Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup TSI_Peripheral_Access_Layer TSI Peripheral Access Layer  
 * @{  
 */
```

```
/** TSI - Register Layout Typedef */  
  
typedef struct {  
  
    __IO uint32_t GENCS;           /**< TSI General Control and Status Register, offset: 0x0 */  
    __IO uint32_t DATA;            /**< TSI DATA Register, offset: 0x4 */  
    __IO uint32_t TSHD;            /**< TSI Threshold Register, offset: 0x8 */  
  
} TSI_Type, *TSI_MemMapPtr;
```

```
/* -----  
-- TSI - Register accessor macros  
----- */
```

```
/*!  
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros  
 * @{
```

```
 */
```

```
/* TSI - Register accessors */

#define TSI_GENCS_REG(base)          ((base)->GENCS)
#define TSI_DATA_REG(base)           ((base)->DATA)
#define TSI_TS HD_REG(base)          ((base)->TS HD)

/*!
 * @}

/* /* end of group TSI_Register_Accessor_Macros */
```

```
/* -----
-- TSI Register Masks
----- */
```

```
/*!
 * @addtogroup TSI_Register_Masks TSI Register Masks
 * @{
 */

/*
```

```
/* GENCS Bit Fields */

#define TSI_GENCS_CURSW_MASK        0x2u
#define TSI_GENCS_CURSW_SHIFT       1
#define TSI_GENCS_CURSW_WIDTH       1

#define TSI_GENCS_CURSW(x)
((uint32_t)((uint32_t)(x)<<TSI_GENCS_CURSW_SHIFT))&TSI_GENCS_CURSW_MASK

#define TSI_GENCS_EOSF_MASK         0x4u
```

```

#define TSI_GENCS_EOSF_SHIFT          2
#define TSI_GENCS_EOSF_WIDTH          1
#define TSI_GENCS_EOSF(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_EOSF_SHIFT))&TSI_GENCS_EOSF_MASK)

#define TSI_GENCS_SCNIP_MASK          0x8u
#define TSI_GENCS_SCNIP_SHIFT         3
#define TSI_GENCS_SCNIP_WIDTH         1
#define TSI_GENCS_SCNIP(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_SCNIP_SHIFT))&TSI_GENCS_SCNIP_MASK)

#define TSI_GENCS_STM_MASK            0x10u
#define TSI_GENCS_STM_SHIFT           4
#define TSI_GENCS_STM_WIDTH           1
#define TSI_GENCS_STM(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_STM_SHIFT))&TSI_GENCS_STM_MASK)

#define TSI_GENCS_STPE_MASK           0x20u
#define TSI_GENCS_STPE_SHIFT          5
#define TSI_GENCS_STPE_WIDTH          1
#define TSI_GENCS_STPE(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_STPE_SHIFT))&TSI_GENCS_STPE_MASK)

#define TSI_GENCS_TSIIEN_MASK          0x40u
#define TSI_GENCS_TSIIEN_SHIFT         6
#define TSI_GENCS_TSIIEN_WIDTH         1
#define TSI_GENCS_TSIIEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIIEN_SHIFT))&TSI_GENCS_TSIIEN_MASK)

#define TSI_GENCS_TSIEN_MASK           0x80u
#define TSI_GENCS_TSIEN_SHIFT          7
#define TSI_GENCS_TSIEN_WIDTH          1
#define TSI_GENCS_TSIEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_TSIEN_SHIFT))&TSI_GENCS_TSIEN_MASK)

#define TSI_GENCS_NSCN_MASK            0x1FO0u
#define TSI_GENCS_NSCN_SHIFT           8

```

```

#define TSI_GENCS_NSCN_WIDTH          5
#define TSI_GENCS_NSCN(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_NSCN_SHIFT))&TSI_GENCS_NSCN_MASK)

#define TSI_GENCS_PS_MASK            0xE000u
#define TSI_GENCS_PS_SHIFT           13
#define TSI_GENCS_PS_WIDTH           3
#define TSI_GENCS_PS(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_PS_SHIFT))&TSI_GENCS_PS_MASK)

#define TSI_GENCS_EXTCHRG_MASK       0x70000u
#define TSI_GENCS_EXTCHRG_SHIFT      16
#define TSI_GENCS_EXTCHRG_WIDTH      3
#define TSI_GENCS_EXTCHRG(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_EXTCHRG_SHIFT))&TSI_GENCS_EXTCHRG_MASK)

#define TSI_GENCS_DVOLT_MASK         0x180000u
#define TSI_GENCS_DVOLT_SHIFT        19
#define TSI_GENCS_DVOLT_WIDTH        2
#define TSI_GENCS_DVOLT(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_DVOLT_SHIFT))&TSI_GENCS_DVOLT_MASK)

#define TSI_GENCS_REFCHRG_MASK       0xE00000u
#define TSI_GENCS_REFCHRG_SHIFT      21
#define TSI_GENCS_REFCHRG_WIDTH      3
#define TSI_GENCS_REFCHRG(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_REFCHRG_SHIFT))&TSI_GENCS_REFCHRG_MASK)

#define TSI_GENCS_MODE_MASK          0xF000000u
#define TSI_GENCS_MODE_SHIFT         24
#define TSI_GENCS_MODE_WIDTH         4
#define TSI_GENCS_MODE(x) (((uint32_t)((uint32_t)(x)<<TSI_GENCS_MODE_SHIFT))&TSI_GENCS_MODE_MASK)

#define TSI_GENCS_ESOR_MASK          0x10000000u
#define TSI_GENCS_ESOR_SHIFT         28
#define TSI_GENCS_ESOR_WIDTH         1

```

```

#define TSI_GENCS_ESOR(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_ESOR_SHIFT))&TSI_GENCS_ESOR_MASK)

#define TSI_GENCS_OUTRGF_MASK          0x80000000u
#define TSI_GENCS_OUTRGF_SHIFT         31
#define TSI_GENCS_OUTRGF_WIDTH         1

#define TSI_GENCS_OUTRGF(x)
(((uint32_t)((uint32_t)(x)<<TSI_GENCS_OUTRGF_SHIFT))&TSI_GENCS_OUTRGF_MASK)

/* DATA Bit Fields */

#define TSI_DATA_TSICNT_MASK          0xFFFFu
#define TSI_DATA_TSICNT_SHIFT         0
#define TSI_DATA_TSICNT_WIDTH         16

#define TSI_DATA_TSICNT(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICNT_SHIFT))&TSI_DATA_TSICNT_MASK)

#define TSI_DATA_SWTS_MASK            0x400000u
#define TSI_DATA_SWTS_SHIFT           22
#define TSI_DATA_SWTS_WIDTH           1

#define TSI_DATA_SWTS(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_SWTS_SHIFT))&TSI_DATA_SWTS_MASK)

#define TSI_DATA_DMAEN_MASK           0x800000u
#define TSI_DATA_DMAEN_SHIFT          23
#define TSI_DATA_DMAEN_WIDTH          1

#define TSI_DATA_DMAEN(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_DMAEN_SHIFT))&TSI_DATA_DMAEN_MASK)

#define TSI_DATA_TSICH_MASK           0xF0000000u
#define TSI_DATA_TSICH_SHIFT          28
#define TSI_DATA_TSICH_WIDTH          4

#define TSI_DATA_TSICH(x)
(((uint32_t)((uint32_t)(x)<<TSI_DATA_TSICH_SHIFT))&TSI_DATA_TSICH_MASK)

/* TSHD Bit Fields */

#define TSI_TSHD_THRESL_MASK          0xFFFFu
#define TSI_TSHD_THRESL_SHIFT          0

```

```

#define TSI_TS HD_THRESL_WIDTH      16
#define TSI_TS HD_THRESL(x)
(((uint32_t) (((uint32_t)(x))<<TSI_TS HD_THRESL_SHIFT))&TSI_TS HD_THRESL_MASK)

#define TSI_TS HD_THRESH_MASK        0xFFFF0000u
#define TSI_TS HD_THRESH_SHIFT      16
#define TSI_TS HD_THRESH_WIDTH      16
#define TSI_TS HD_THRESH(x)
(((uint32_t) (((uint32_t)(x))<<TSI_TS HD_THRESH_SHIFT))&TSI_TS HD_THRESH_MASK)

/*
 * @}
 */

/* /* end of group TSI_Register_Masks */

/* TSI - Peripheral instance base addresses */
/** Peripheral TSI0 base address */
#define TSI0_BASE                  (0x40045000u)
/** Peripheral TSI0 base pointer */
#define TSI0 ((TSI_Type *)TSI0_BASE)
#define TSI0_BASE_PTR              (TSI0)
/** Array initializer of TSI peripheral base addresses */
#define TSI_BASE_ADDRS             { TSI0_BASE }
/** Array initializer of TSI peripheral base pointers */
#define TSI_BASE_PTRS              { TSI0 }

/*
-----*
-- TSI - Register accessor macros
-----*/

```

```
/*!  
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor macros  
 * @{  
 */
```

```
/* TSI - Register instance definitions */  
/* TSI0 */  
  
#define TSI0_GENCS          TSI_GENCS_REG(TSI0)  
#define TSI0_DATA           TSI_DATA_REG(TSI0)  
#define TSI0_TS HD          TSI_TS HD_REG(TSI0)
```

```
/*!  
 * @}  
 */ /* end of group TSI_Register_Accessor_Macros */
```

```
/*!  
 * @}  
 */ /* end of group TSI_Peripheral_Access_Layer */
```

```
/* -----  
-- UART Peripheral Access Layer  
----- */
```

```
/*!  
 * @addtogroup UART_Peripheral_Access_Layer UART Peripheral Access Layer  
 * @{
```

```

*/



/** UART - Register Layout Typedef */

typedef struct {

    __IO uint8_t BDH;           /**< UART Baud Rate Register: High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register: Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __I  uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;              /**< UART Data Register, offset: 0x7 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0x8 */

} UART_Type, *UART_MemMapPtr;

```

```

/*
-----*
-- UART - Register accessor macros
-----*/

```

```

/*!
* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros
* @{
*/

```

```

/* UART - Register accessors */

#define UART_BDH_REG(base)      ((base)->BDH)
#define UART_BDL_REG(base)      ((base)->BDL)
#define UART_C1_REG(base)       ((base)->C1)

```

```
#define UART_C2_REG(base)          ((base)->C2)
#define UART_S1_REG(base)           ((base)->S1)
#define UART_S2_REG(base)           ((base)->S2)
#define UART_C3_REG(base)           ((base)->C3)
#define UART_D_REG(base)            ((base)->D)
#define UART_C4_REG(base)           ((base)->C4)
```

```
/*!
 * @}
 */
/* /* end of group UART_Register_Accessor_Macros */
```

```
/*
-----  
-- UART Register Masks  
----- */
```

```
/*!
 * @addtogroup UART_Register_Masks UART Register Masks
 * @{
 */

```

```
/* BDH Bit Fields */

#define UART_BDH_SBR_MASK          0x1Fu
#define UART_BDH_SBR_SHIFT         0
#define UART_BDH_SBR_WIDTH          5

#define UART_BDH_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART_BDH_SBR_SHIFT))&UART_BDH_SBR_MASK)

#define UART_BDH_SBNS_MASK         0x20u
#define UART_BDH_SBNS_SHIFT          5
```

```

#define UART_BDH_SBNS_WIDTH          1
#define UART_BDH_SBNS(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_SBNS_SHIFT))&UART_BDH_SBNS_MASK)
#define UART_BDH_RXEDGIE_MASK        0x40u
#define UART_BDH_RXEDGIE_SHIFT       6
#define UART_BDH_RXEDGIE_WIDTH       1
#define UART_BDH_RXEDGIE(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_RXEDGIE_SHIFT))&UART_BDH_RXEDGIE_MASK)
#define UART_BDH_LBKDIE_MASK         0x80u
#define UART_BDH_LBKDIE_SHIFT        7
#define UART_BDH_LBKDIE_WIDTH        1
#define UART_BDH_LBKDIE(x) (((uint8_t)((uint8_t)(x)<<UART_BDH_LBKDIE_SHIFT))&UART_BDH_LBKDIE_MASK)

/* BDL Bit Fields */

#define UART_BDL_SBR_MASK           0xFFu
#define UART_BDL_SBR_SHIFT          0
#define UART_BDL_SBR_WIDTH          8
#define UART_BDL_SBR(x) (((uint8_t)((uint8_t)(x)<<UART_BDL_SBR_SHIFT))&UART_BDL_SBR_MASK)

/* C1 Bit Fields */

#define UART_C1_PT_MASK             0x1u
#define UART_C1_PT_SHIFT             0
#define UART_C1_PT_WIDTH             1
#define UART_C1_PT(x) (((uint8_t)((uint8_t)(x)<<UART_C1_PT_SHIFT))&UART_C1_PT_MASK)
#define UART_C1_PE_MASK              0x2u
#define UART_C1_PE_SHIFT             1
#define UART_C1_PE_WIDTH             1
#define UART_C1_PE(x) (((uint8_t)((uint8_t)(x)<<UART_C1_PE_SHIFT))&UART_C1_PE_MASK)
#define UART_C1_ILT_MASK             0x4u

```

```

#define UART_C1_ILT_SHIFT          2
#define UART_C1_ILT_WIDTH          1
#define UART_C1_ILT(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_ILT_SHIFT))&UART_C1_ILT_MASK)

#define UART_C1_WAKE_MASK          0x8u
#define UART_C1_WAKE_SHIFT          3
#define UART_C1_WAKE_WIDTH          1
#define UART_C1_WAKE(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_WAKE_SHIFT))&UART_C1_WAKE_MASK)

#define UART_C1_M_MASK              0x10u
#define UART_C1_M_SHIFT             4
#define UART_C1_M_WIDTH              1
#define UART_C1_M(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_M_SHIFT))&UART_C1_M_MASK)

#define UART_C1_RSRC_MASK           0x20u
#define UART_C1_RSRC_SHIFT           5
#define UART_C1_RSRC_WIDTH           1
#define UART_C1_RSRC(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_RSRC_SHIFT))&UART_C1_RSRC_MASK)

#define UART_C1_UARTSWAI_MASK        0x40u
#define UART_C1_UARTSWAI_SHIFT        6
#define UART_C1_UARTSWAI_WIDTH        1
#define UART_C1_UARTSWAI(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_UARTSWAI_SHIFT))&UART_C1_UARTSWAI_MASK)

#define UART_C1_LOOPS_MASK           0x80u
#define UART_C1_LOOPS_SHIFT           7
#define UART_C1_LOOPS_WIDTH           1
#define UART_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x)<<UART_C1_LOOPS_SHIFT))&UART_C1_LOOPS_MASK)

/* C2 Bit Fields */

#define UART_C2_SBK_MASK             0x1u

```

```

#define UART_C2_SBK_SHIFT          0
#define UART_C2_SBK_WIDTH          1
#define UART_C2_SBK(x)             (((uint8_t)((uint8_t)(x)<<UART_C2_SBK_SHIFT))&UART_C2_SBK_MASK)

#define UART_C2_RWU_MASK           0x2u
#define UART_C2_RWU_SHIFT          1
#define UART_C2_RWU_WIDTH          1
#define UART_C2_RWU(x)             (((uint8_t)((uint8_t)(x)<<UART_C2_RWU_SHIFT))&UART_C2_RWU_MASK)

#define UART_C2_RE_MASK            0x4u
#define UART_C2_RE_SHIFT           2
#define UART_C2_RE_WIDTH           1
#define UART_C2_RE(x)              (((uint8_t)((uint8_t)(x)<<UART_C2_RE_SHIFT))&UART_C2_RE_MASK)

#define UART_C2_TE_MASK            0x8u
#define UART_C2_TE_SHIFT           3
#define UART_C2_TE_WIDTH           1
#define UART_C2_TE(x)              (((uint8_t)((uint8_t)(x)<<UART_C2_TE_SHIFT))&UART_C2_TE_MASK)

#define UART_C2_ILIE_MASK          0x10u
#define UART_C2_ILIE_SHIFT          4
#define UART_C2_ILIE_WIDTH          1
#define UART_C2_ILIE(x)             (((uint8_t)((uint8_t)(x)<<UART_C2_ILIE_SHIFT))&UART_C2_ILIE_MASK)

#define UART_C2_RIE_MASK           0x20u
#define UART_C2_RIE_SHIFT          5
#define UART_C2_RIE_WIDTH           1
#define UART_C2_RIE(x)              (((uint8_t)((uint8_t)(x)<<UART_C2_RIE_SHIFT))&UART_C2_RIE_MASK)

#define UART_C2_TCIE_MASK          0x40u
#define UART_C2_TCIE_SHIFT          6

```

```

#define UART_C2_TCIE_WIDTH           1
#define UART_C2_TCIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TCIE_SHIFT))&UART_C2_TCIE_MASK)
#define UART_C2_TIE_MASK             0x80u
#define UART_C2_TIE_SHIFT            7
#define UART_C2_TIE_WIDTH            1
#define UART_C2_TIE(x) (((uint8_t)((uint8_t)(x)<<UART_C2_TIE_SHIFT))&UART_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART_S1_PF_MASK              0x1u
#define UART_S1_PF_SHIFT              0
#define UART_S1_PF_WIDTH              1
#define UART_S1_PF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_PF_SHIFT))&UART_S1_PF_MASK)

#define UART_S1_FE_MASK              0x2u
#define UART_S1_FE_SHIFT              1
#define UART_S1_FE_WIDTH              1
#define UART_S1_FE(x) (((uint8_t)((uint8_t)(x)<<UART_S1_FE_SHIFT))&UART_S1_FE_MASK)

#define UART_S1_NF_MASK              0x4u
#define UART_S1_NF_SHIFT              2
#define UART_S1_NF_WIDTH              1
#define UART_S1_NF(x) (((uint8_t)((uint8_t)(x)<<UART_S1_NF_SHIFT))&UART_S1_NF_MASK)

#define UART_S1_OR_MASK              0x8u
#define UART_S1_OR_SHIFT              3
#define UART_S1_OR_WIDTH              1
#define UART_S1_OR(x) (((uint8_t)((uint8_t)(x)<<UART_S1_OR_SHIFT))&UART_S1_OR_MASK)

#define UART_S1_IDLE_MASK             0x10u
#define UART_S1_IDLE_SHIFT             4

```

```

#define UART_S1_IDLE_WIDTH          1
#define UART_S1_IDLE(x)             (((uint8_t)((uint8_t)(x)<<UART_S1_IDLE_SHIFT))&UART_S1_IDLE_MASK)
#define UART_S1_RDRF_MASK           0x20u
#define UART_S1_RDRF_SHIFT          5
#define UART_S1_RDRF_WIDTH          1
#define UART_S1_RDRF(x)             (((uint8_t)((uint8_t)(x)<<UART_S1_RDRF_SHIFT))&UART_S1_RDRF_MASK)
#define UART_S1_TC_MASK              0x40u
#define UART_S1_TC_SHIFT             6
#define UART_S1_TC_WIDTH             1
#define UART_S1_TC(x)               (((uint8_t)((uint8_t)(x)<<UART_S1_TC_SHIFT))&UART_S1_TC_MASK)
#define UART_S1_TDRE_MASK            0x80u
#define UART_S1_TDRE_SHIFT           7
#define UART_S1_TDRE_WIDTH           1
#define UART_S1_TDRE(x)              (((uint8_t)((uint8_t)(x)<<UART_S1_TDRE_SHIFT))&UART_S1_TDRE_MASK)

/* S2 Bit Fields */

#define UART_S2_RAF_MASK             0x1u
#define UART_S2_RAF_SHIFT             0
#define UART_S2_RAF_WIDTH             1
#define UART_S2_RAF(x)               (((uint8_t)((uint8_t)(x)<<UART_S2_RAF_SHIFT))&UART_S2_RAF_MASK)

#define UART_S2_LBKDE_MASK            0x2u
#define UART_S2_LBKDE_SHIFT           1
#define UART_S2_LBKDE_WIDTH           1
#define UART_S2_LBKDE(x)              (((uint8_t)((uint8_t)(x)<<UART_S2_LBKDE_SHIFT))&UART_S2_LBKDE_MASK)

#define UART_S2_BRK13_MASK             0x4u
#define UART_S2_BRK13_SHIFT            2

```

```

#define UART_S2_BRK13_WIDTH          1
#define UART_S2_BRK13(x) (((uint8_t)((uint8_t)(x)<<UART_S2_BRK13_SHIFT))&UART_S2_BRK13_MASK)
#define UART_S2_RWUID_MASK           0x8u
#define UART_S2_RWUID_SHIFT          3
#define UART_S2_RWUID_WIDTH          1
#define UART_S2_RWUID(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RWUID_SHIFT))&UART_S2_RWUID_MASK)
#define UART_S2_RXINV_MASK           0x10u
#define UART_S2_RXINV_SHIFT          4
#define UART_S2_RXINV_WIDTH          1
#define UART_S2_RXINV(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RXINV_SHIFT))&UART_S2_RXINV_MASK)
#define UART_S2_RXEDGIF_MASK          0x40u
#define UART_S2_RXEDGIF_SHIFT         6
#define UART_S2_RXEDGIF_WIDTH         1
#define UART_S2_RXEDGIF(x) (((uint8_t)((uint8_t)(x)<<UART_S2_RXEDGIF_SHIFT))&UART_S2_RXEDGIF_MASK)
#define UART_S2_LBKDIF_MASK           0x80u
#define UART_S2_LBKDIF_SHIFT          7
#define UART_S2_LBKDIF_WIDTH          1
#define UART_S2_LBKDIF(x) (((uint8_t)((uint8_t)(x)<<UART_S2_LBKDIF_SHIFT))&UART_S2_LBKDIF_MASK)
/* C3 Bit Fields */
#define UART_C3_PEIE_MASK            0x1u
#define UART_C3_PEIE_SHIFT             0
#define UART_C3_PEIE_WIDTH             1
#define UART_C3_PEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_PEIE_SHIFT))&UART_C3_PEIE_MASK)
#define UART_C3_FEIE_MASK             0x2u
#define UART_C3_FEIE_SHIFT              1

```

```

#define UART_C3_FEIE_WIDTH          1
#define UART_C3_FEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_FEIE_SHIFT))&UART_C3_FEIE_MASK)
#define UART_C3_NEIE_MASK           0x4u
#define UART_C3_NEIE_SHIFT          2
#define UART_C3_NEIE_WIDTH          1
#define UART_C3_NEIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_NEIE_SHIFT))&UART_C3_NEIE_MASK)
#define UART_C3_ORIE_MASK           0x8u
#define UART_C3_ORIE_SHIFT          3
#define UART_C3_ORIE_WIDTH          1
#define UART_C3_ORIE(x) (((uint8_t)((uint8_t)(x)<<UART_C3_ORIE_SHIFT))&UART_C3_ORIE_MASK)
#define UART_C3_TXINV_MASK           0x10u
#define UART_C3_TXINV_SHIFT          4
#define UART_C3_TXINV_WIDTH          1
#define UART_C3_TXINV(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXINV_SHIFT))&UART_C3_TXINV_MASK)
#define UART_C3_TXDIR_MASK           0x20u
#define UART_C3_TXDIR_SHIFT          5
#define UART_C3_TXDIR_WIDTH          1
#define UART_C3_TXDIR(x) (((uint8_t)((uint8_t)(x)<<UART_C3_TXDIR_SHIFT))&UART_C3_TXDIR_MASK)
#define UART_C3_T8_MASK              0x40u
#define UART_C3_T8_SHIFT             6
#define UART_C3_T8_WIDTH             1
#define UART_C3_T8(x) (((uint8_t)((uint8_t)(x)<<UART_C3_T8_SHIFT))&UART_C3_T8_MASK)
#define UART_C3_R8_MASK              0x80u
#define UART_C3_R8_SHIFT             7
#define UART_C3_R8_WIDTH             1

```

```

#define UART_C3_R8(x)
(((uint8_t)((uint8_t)(x)<<UART_C3_R8_SHIFT))&UART_C3_R8_MASK)

/* D Bit Fields */

#define UART_DROTO_MASK          0x1u
#define UART_DROTO_SHIFT         0
#define UART_DROTO_WIDTH         1

#define UART_DROTO(x)
(((uint8_t)((uint8_t)(x)<<UART_DROTO_SHIFT))&UART_DROTO_MASK)

#define UART_D_R1T1_MASK          0x2u
#define UART_D_R1T1_SHIFT         1
#define UART_D_R1T1_WIDTH         1

#define UART_D_R1T1(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R1T1_SHIFT))&UART_D_R1T1_MASK)

#define UART_D_R2T2_MASK          0x4u
#define UART_D_R2T2_SHIFT         2
#define UART_D_R2T2_WIDTH         1

#define UART_D_R2T2(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R2T2_SHIFT))&UART_D_R2T2_MASK)

#define UART_D_R3T3_MASK          0x8u
#define UART_D_R3T3_SHIFT         3
#define UART_D_R3T3_WIDTH         1

#define UART_D_R3T3(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R3T3_SHIFT))&UART_D_R3T3_MASK)

#define UART_D_R4T4_MASK          0x10u
#define UART_D_R4T4_SHIFT         4
#define UART_D_R4T4_WIDTH         1

#define UART_D_R4T4(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R4T4_SHIFT))&UART_D_R4T4_MASK)

#define UART_D_R5T5_MASK          0x20u
#define UART_D_R5T5_SHIFT         5
#define UART_D_R5T5_WIDTH         1

```

```

#define UART_D_R5T5(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R5T5_SHIFT))&UART_D_R5T5_MASK)

#define UART_D_R6T6_MASK          0x40u
#define UART_D_R6T6_SHIFT         6
#define UART_D_R6T6_WIDTH         1

#define UART_D_R6T6(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R6T6_SHIFT))&UART_D_R6T6_MASK)

#define UART_D_R7T7_MASK          0x80u
#define UART_D_R7T7_SHIFT         7
#define UART_D_R7T7_WIDTH         1

#define UART_D_R7T7(x)
(((uint8_t)((uint8_t)(x)<<UART_D_R7T7_SHIFT))&UART_D_R7T7_MASK)

/* C4 Bit Fields */

#define UART_C4_RDMAS_MASK        0x20u
#define UART_C4_RDMAS_SHIFT       5
#define UART_C4_RDMAS_WIDTH       1

#define UART_C4_RDMAS(x)
(((uint8_t)((uint8_t)(x)<<UART_C4_RDMAS_SHIFT))&UART_C4_RDMAS_MASK)

#define UART_C4_TDMAS_MASK        0x80u
#define UART_C4_TDMAS_SHIFT       7
#define UART_C4_TDMAS_WIDTH       1

#define UART_C4_TDMAS(x)
(((uint8_t)((uint8_t)(x)<<UART_C4_TDMAS_SHIFT))&UART_C4_TDMAS_MASK)

/*
 * @}
 */

/* /* end of group UART_Register_Masks */

/* UART - Peripheral instance base addresses */

```

```

/** Peripheral UART1 base address */
#define UART1_BASE           (0x4006B000u)

/** Peripheral UART1 base pointer */
#define UART1             ((UART_Type *)UART1_BASE)
#define UART1_BASE_PTR     (UART1)

/** Peripheral UART2 base address */
#define UART2_BASE           (0x4006C000u)

/** Peripheral UART2 base pointer */
#define UART2             ((UART_Type *)UART2_BASE)
#define UART2_BASE_PTR     (UART2)

/** Array initializer of UART peripheral base addresses */
#define UART_BASE_ADDRS      { UART1_BASE, UART2_BASE }

/** Array initializer of UART peripheral base pointers */
#define UART_BASE_PTRS       { UART1, UART2 }

```

```

/*
-----  

-- UART - Register accessor macros  

----- */

```

```

/*!  

* @addtogroup UART_Register_Accessor_Macros UART - Register accessor macros  

* @{
* /  


```

```

/* UART - Register instance definitions */  

/* UART1 */  

#define UART1_BDH           UART_BDH_REG(UART1)  

#define UART1_BDL           UART_BDL_REG(UART1)

```

```
#define UART1_C1          UART_C1_REG(UART1)
#define UART1_C2          UART_C2_REG(UART1)
#define UART1_S1          UART_S1_REG(UART1)
#define UART1_S2          UART_S2_REG(UART1)
#define UART1_C3          UART_C3_REG(UART1)
#define UART1_D           UART_D_REG(UART1)
#define UART1_C4          UART_C4_REG(UART1)

/* UART2 */

#define UART2_BDH          UART_BDH_REG(UART2)
#define UART2_BDL          UART_BDL_REG(UART2)
#define UART2_C1          UART_C1_REG(UART2)
#define UART2_C2          UART_C2_REG(UART2)
#define UART2_S1          UART_S1_REG(UART2)
#define UART2_S2          UART_S2_REG(UART2)
#define UART2_C3          UART_C3_REG(UART2)
#define UART2_D           UART_D_REG(UART2)
#define UART2_C4          UART_C4_REG(UART2)

/*!
 * @}
 */
/* /* end of group UART_Register_Accessor_Macros */
```

```
/*
 * @}
 */
/* /* end of group UART_Peripheral_Access_Layer */
```

```
/* -----
```

```

-- UART0 Peripheral Access Layer
----- */
/*!
* @addtogroup UART0_Peripheral_Access_Layer UART0 Peripheral Access Layer
* @{
*/
/** UART0 - Register Layout Typedef */
typedef struct {
    __IO uint8_t BDH;           /**< UART Baud Rate Register High, offset: 0x0 */
    __IO uint8_t BDL;           /**< UART Baud Rate Register Low, offset: 0x1 */
    __IO uint8_t C1;            /**< UART Control Register 1, offset: 0x2 */
    __IO uint8_t C2;            /**< UART Control Register 2, offset: 0x3 */
    __IO uint8_t S1;            /**< UART Status Register 1, offset: 0x4 */
    __IO uint8_t S2;            /**< UART Status Register 2, offset: 0x5 */
    __IO uint8_t C3;            /**< UART Control Register 3, offset: 0x6 */
    __IO uint8_t D;              /**< UART Data Register, offset: 0x7 */
    __IO uint8_t MA1;           /**< UART Match Address Registers 1, offset: 0x8 */
    __IO uint8_t MA2;           /**< UART Match Address Registers 2, offset: 0x9 */
    __IO uint8_t C4;            /**< UART Control Register 4, offset: 0xA */
    __IO uint8_t C5;            /**< UART Control Register 5, offset: 0xB */
} UART0_Type, *UART0_MemMapPtr;

/*
----- --
-- UART0 - Register accessor macros
----- */
/*!

```

```
* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros
* @{
* /
```

```
/* UART0 - Register accessors */

#define UART0_BDH_REG(base)          ((base)->BDH)
#define UART0_BDL_REG(base)          ((base)->BDL)
#define UART0_C1_REG(base)           ((base)->C1)
#define UART0_C2_REG(base)           ((base)->C2)
#define UART0_S1_REG(base)           ((base)->S1)
#define UART0_S2_REG(base)           ((base)->S2)
#define UART0_C3_REG(base)           ((base)->C3)
#define UART0_D_REG(base)            ((base)->D)
#define UART0_MA1_REG(base)          ((base)->MA1)
#define UART0_MA2_REG(base)          ((base)->MA2)
#define UART0_C4_REG(base)           ((base)->C4)
#define UART0_C5_REG(base)           ((base)->C5)
```

```
/*
* @@
* /* end of group UART0_Register_Accessor_Macros */
```

```
/* -----
-- UART0 Register Masks
----- */
```

```
/*!
```

```

* @addtogroup UART0_Register_Masks UART0 Register Masks

* @{
*

/* BDH Bit Fields */

#define UART0_BDH_SBR_MASK          0x1Fu
#define UART0_BDH_SBR_SHIFT         0
#define UART0_BDH_SBR_WIDTH         5
#define UART0_BDH_SBR(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_SBR_SHIFT))&UART0_BDH_SBR_MASK)

#define UART0_BDH_SBNS_MASK         0x20u
#define UART0_BDH_SBNS_SHIFT        5
#define UART0_BDH_SBNS_WIDTH        1
#define UART0_BDH_SBNS(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_SBNS_SHIFT))&UART0_BDH_SBNS_MASK)

#define UART0_BDH_RXEDGIE_MASK       0x40u
#define UART0_BDH_RXEDGIE_SHIFT      6
#define UART0_BDH_RXEDGIE_WIDTH      1
#define UART0_BDH_RXEDGIE(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_RXEDGIE_SHIFT))&UART0_BDH_RXEDGIE_MASK)

#define UART0_BDH_LBKDIE_MASK        0x80u
#define UART0_BDH_LBKDIE_SHIFT       7
#define UART0_BDH_LBKDIE_WIDTH       1
#define UART0_BDH_LBKDIE(x) (((uint8_t)((uint8_t)(x)<<UART0_BDH_LBKDIE_SHIFT))&UART0_BDH_LBKDIE_MASK)

/* BDL Bit Fields */

#define UART0_BDL_SBR_MASK          0xFFu
#define UART0_BDL_SBR_SHIFT         0
#define UART0_BDL_SBR_WIDTH         8

```

```

#define UART0_BDL_SBR(x)
(((uint8_t)((uint8_t)(x)<<UART0_BDL_SBR_SHIFT))&UART0_BDL_SBR_MASK)

/* C1 Bit Fields */

#define UART0_C1_PT_MASK          0x1u
#define UART0_C1_PT_SHIFT         0
#define UART0_C1_PT_WIDTH         1

#define UART0_C1_PT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PT_SHIFT))&UART0_C1_PT_MASK)

#define UART0_C1_PE_MASK          0x2u
#define UART0_C1_PE_SHIFT         1
#define UART0_C1_PE_WIDTH         1

#define UART0_C1_PE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_PE_SHIFT))&UART0_C1_PE_MASK)

#define UART0_C1_ILT_MASK          0x4u
#define UART0_C1_ILT_SHIFT         2
#define UART0_C1_ILT_WIDTH         1

#define UART0_C1_ILT(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_ILT_SHIFT))&UART0_C1_ILT_MASK)

#define UART0_C1_WAKE_MASK          0x8u
#define UART0_C1_WAKE_SHIFT         3
#define UART0_C1_WAKE_WIDTH         1

#define UART0_C1_WAKE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_WAKE_SHIFT))&UART0_C1_WAKE_MASK)

#define UART0_C1_M_MASK           0x10u
#define UART0_C1_M_SHIFT            4
#define UART0_C1_M_WIDTH             1

#define UART0_C1_M(x)
(((uint8_t)((uint8_t)(x)<<UART0_C1_M_SHIFT))&UART0_C1_M_MASK)

#define UART0_C1_RSRC_MASK          0x20u
#define UART0_C1_RSRC_SHIFT         5
#define UART0_C1_RSRC_WIDTH         1

```

```

#define UART0_C1_RSRC(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_RSRC_SHIFT))&UART0_C1_RSRC_MASK)

#define UART0_C1_DOZEEN_MASK          0x40u
#define UART0_C1_DOZEEN_SHIFT        6
#define UART0_C1_DOZEEN_WIDTH        1

#define UART0_C1_DOZEEN(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_DOZEEN_SHIFT))&UART0_C1_DOZEEN_MASK)

#define UART0_C1_LOOPS_MASK          0x80u
#define UART0_C1_LOOPS_SHIFT        7
#define UART0_C1_LOOPS_WIDTH        1

#define UART0_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x))<<UART0_C1_LOOPS_SHIFT))&UART0_C1_LOOPS_MASK)

/* C2 Bit Fields */

#define UART0_C2_SBK_MASK           0x1u
#define UART0_C2_SBK_SHIFT          0
#define UART0_C2_SBK_WIDTH          1

#define UART0_C2_SBK(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_SBK_SHIFT))&UART0_C2_SBK_MASK)

#define UART0_C2_RWU_MASK           0x2u
#define UART0_C2_RWU_SHIFT          1
#define UART0_C2_RWU_WIDTH          1

#define UART0_C2_RWU(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RWU_SHIFT))&UART0_C2_RWU_MASK)

#define UART0_C2_RE_MASK            0x4u
#define UART0_C2_RE_SHIFT           2
#define UART0_C2_RE_WIDTH           1

#define UART0_C2_RE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C2_RE_SHIFT))&UART0_C2_RE_MASK)

#define UART0_C2_TE_MASK            0x8u
#define UART0_C2_TE_SHIFT           3
#define UART0_C2_TE_WIDTH           1

```

```

#define UART0_C2_TE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TE_SHIFT))&UART0_C2_TE_MASK)

#define UART0_C2_ILIE_MASK          0x10u
#define UART0_C2_ILIE_SHIFT         4
#define UART0_C2_ILIE_WIDTH         1

#define UART0_C2_ILIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_ILIE_SHIFT))&UART0_C2_ILIE_MASK)

#define UART0_C2_RIE_MASK           0x20u
#define UART0_C2_RIE_SHIFT          5
#define UART0_C2_RIE_WIDTH          1

#define UART0_C2_RIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_RIE_SHIFT))&UART0_C2_RIE_MASK)

#define UART0_C2_TCIE_MASK          0x40u
#define UART0_C2_TCIE_SHIFT          6
#define UART0_C2_TCIE_WIDTH          1

#define UART0_C2_TCIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TCIE_SHIFT))&UART0_C2_TCIE_MASK)

#define UART0_C2_TIE_MASK            0x80u
#define UART0_C2_TIE_SHIFT           7
#define UART0_C2_TIE_WIDTH           1

#define UART0_C2_TIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C2_TIE_SHIFT))&UART0_C2_TIE_MASK)

/* S1 Bit Fields */

#define UART0_S1_PF_MASK             0x1u
#define UART0_S1_PF_SHIFT              0
#define UART0_S1_PF_WIDTH              1

#define UART0_S1_PF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_PF_SHIFT))&UART0_S1_PF_MASK)

#define UART0_S1_FE_MASK              0x2u
#define UART0_S1_FE_SHIFT                1
#define UART0_S1_FE_WIDTH                1

```

```

#define UART0_S1_FE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_FE_SHIFT))&UART0_S1_FE_MASK)

#define UART0_S1_NF_MASK          0x4u
#define UART0_S1_NF_SHIFT         2
#define UART0_S1_NF_WIDTH         1

#define UART0_S1_NF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_NF_SHIFT))&UART0_S1_NF_MASK)

#define UART0_S1_OR_MASK          0x8u
#define UART0_S1_OR_SHIFT         3
#define UART0_S1_OR_WIDTH         1

#define UART0_S1_OR(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_OR_SHIFT))&UART0_S1_OR_MASK)

#define UART0_S1_IDLE_MASK         0x10u
#define UART0_S1_IDLE_SHIFT        4
#define UART0_S1_IDLE_WIDTH        1

#define UART0_S1_IDLE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_IDLE_SHIFT))&UART0_S1_IDLE_MASK)

#define UART0_S1_RDRF_MASK         0x20u
#define UART0_S1_RDRF_SHIFT        5
#define UART0_S1_RDRF_WIDTH        1

#define UART0_S1_RDRF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_RDRF_SHIFT))&UART0_S1_RDRF_MASK)

#define UART0_S1_TC_MASK           0x40u
#define UART0_S1_TC_SHIFT          6
#define UART0_S1_TC_WIDTH          1

#define UART0_S1_TC(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TC_SHIFT))&UART0_S1_TC_MASK)

#define UART0_S1_TDRE_MASK          0x80u
#define UART0_S1_TDRE_SHIFT         7
#define UART0_S1_TDRE_WIDTH         1

```

```

#define UART0_S1_TDRE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S1_TDRE_SHIFT))&UART0_S1_TDRE_MASK)

/* S2 Bit Fields */

#define UART0_S2_RAF_MASK          0x1u
#define UART0_S2_RAF_SHIFT         0
#define UART0_S2_RAF_WIDTH         1

#define UART0_S2_RAF(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RAF_SHIFT))&UART0_S2_RAF_MASK)

#define UART0_S2_LBKDE_MASK        0x2u
#define UART0_S2_LBKDE_SHIFT       1
#define UART0_S2_LBKDE_WIDTH       1

#define UART0_S2_LBKDE(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_LBKDE_SHIFT))&UART0_S2_LBKDE_MASK)

#define UART0_S2_BRK13_MASK         0x4u
#define UART0_S2_BRK13_SHIFT        2
#define UART0_S2_BRK13_WIDTH        1

#define UART0_S2_BRK13(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_BRK13_SHIFT))&UART0_S2_BRK13_MASK)

#define UART0_S2_RWUID_MASK         0x8u
#define UART0_S2_RWUID_SHIFT        3
#define UART0_S2_RWUID_WIDTH        1

#define UART0_S2_RWUID(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RWUID_SHIFT))&UART0_S2_RWUID_MASK)

#define UART0_S2_RXINV_MASK         0x10u
#define UART0_S2_RXINV_SHIFT        4
#define UART0_S2_RXINV_WIDTH        1

#define UART0_S2_RXINV(x)
(((uint8_t)((uint8_t)(x)<<UART0_S2_RXINV_SHIFT))&UART0_S2_RXINV_MASK)

#define UART0_S2_MSBF_MASK          0x20u
#define UART0_S2_MSBF_SHIFT         5
#define UART0_S2_MSBF_WIDTH         1

```

```

#define UART0_S2_MSBF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_MSBF_SHIFT))&UART0_S2_MSBF_MASK

#define UART0_S2_RXEDGIF_MASK          0x40u
#define UART0_S2_RXEDGIF_SHIFT        6
#define UART0_S2_RXEDGIF_WIDTH        1

#define UART0_S2_RXEDGIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_RXEDGIF_SHIFT))&UART0_S2_RXEDGIF_MASK

#define UART0_S2_LBKDIF_MASK          0x80u
#define UART0_S2_LBKDIF_SHIFT        7
#define UART0_S2_LBKDIF_WIDTH        1

#define UART0_S2_LBKDIF(x)
(((uint8_t)((uint8_t)(x))<<UART0_S2_LBKDIF_SHIFT))&UART0_S2_LBKDIF_MASK

/* C3 Bit Fields */

#define UART0_C3_PEIE_MASK           0x1u
#define UART0_C3_PEIE_SHIFT          0
#define UART0_C3_PEIE_WIDTH          1

#define UART0_C3_PEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_PEIE_SHIFT))&UART0_C3_PEIE_MASK

#define UART0_C3_FEIE_MASK           0x2u
#define UART0_C3_FEIE_SHIFT          1
#define UART0_C3_FEIE_WIDTH          1

#define UART0_C3_FEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_FEIE_SHIFT))&UART0_C3_FEIE_MASK

#define UART0_C3_NEIE_MASK           0x4u
#define UART0_C3_NEIE_SHIFT          2
#define UART0_C3_NEIE_WIDTH          1

#define UART0_C3_NEIE(x)
(((uint8_t)((uint8_t)(x))<<UART0_C3_NEIE_SHIFT))&UART0_C3_NEIE_MASK

#define UART0_C3_ORIE_MASK           0x8u
#define UART0_C3_ORIE_SHIFT          3
#define UART0_C3_ORIE_WIDTH          1

```

```

#define UART0_C3_ORIE(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_ORIE_SHIFT))&UART0_C3_ORIE_MASK)

#define UART0_C3_TXINV_MASK          0x10u
#define UART0_C3_TXINV_SHIFT        4
#define UART0_C3_TXINV_WIDTH        1

#define UART0_C3_TXINV(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_TXINV_SHIFT))&UART0_C3_TXINV_MASK)

#define UART0_C3_TXDIR_MASK          0x20u
#define UART0_C3_TXDIR_SHIFT        5
#define UART0_C3_TXDIR_WIDTH        1

#define UART0_C3_TXDIR(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_TXDIR_SHIFT))&UART0_C3_TXDIR_MASK)

#define UART0_C3_R9T8_MASK          0x40u
#define UART0_C3_R9T8_SHIFT         6
#define UART0_C3_R9T8_WIDTH         1

#define UART0_C3_R9T8(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_R9T8_SHIFT))&UART0_C3_R9T8_MASK)

#define UART0_C3_R8T9_MASK          0x80u
#define UART0_C3_R8T9_SHIFT         7
#define UART0_C3_R8T9_WIDTH         1

#define UART0_C3_R8T9(x)
(((uint8_t)((uint8_t)(x)<<UART0_C3_R8T9_SHIFT))&UART0_C3_R8T9_MASK)

/* D Bit Fields */

#define UART0_D_ROT0_MASK           0x1u
#define UART0_D_ROT0_SHIFT          0
#define UART0_D_ROT0_WIDTH          1

#define UART0_D_ROT0(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_ROT0_SHIFT))&UART0_D_ROT0_MASK)

#define UART0_D_R1T1_MASK           0x2u
#define UART0_D_R1T1_SHIFT          1
#define UART0_D_R1T1_WIDTH          1

```

```

#define UART0_D_R1T1(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R1T1_SHIFT))&UART0_D_R1T1_MASK)

#define UART0_D_R2T2_MASK          0x4u
#define UART0_D_R2T2_SHIFT         2
#define UART0_D_R2T2_WIDTH         1

#define UART0_D_R2T2(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R2T2_SHIFT))&UART0_D_R2T2_MASK)

#define UART0_D_R3T3_MASK          0x8u
#define UART0_D_R3T3_SHIFT         3
#define UART0_D_R3T3_WIDTH         1

#define UART0_D_R3T3(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R3T3_SHIFT))&UART0_D_R3T3_MASK)

#define UART0_D_R4T4_MASK          0x10u
#define UART0_D_R4T4_SHIFT         4
#define UART0_D_R4T4_WIDTH         1

#define UART0_D_R4T4(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R4T4_SHIFT))&UART0_D_R4T4_MASK)

#define UART0_D_R5T5_MASK          0x20u
#define UART0_D_R5T5_SHIFT         5
#define UART0_D_R5T5_WIDTH         1

#define UART0_D_R5T5(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R5T5_SHIFT))&UART0_D_R5T5_MASK)

#define UART0_D_R6T6_MASK          0x40u
#define UART0_D_R6T6_SHIFT         6
#define UART0_D_R6T6_WIDTH         1

#define UART0_D_R6T6(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R6T6_SHIFT))&UART0_D_R6T6_MASK)

#define UART0_D_R7T7_MASK          0x80u
#define UART0_D_R7T7_SHIFT         7
#define UART0_D_R7T7_WIDTH         1

```

```

#define UART0_D_R7T7(x)
(((uint8_t)((uint8_t)(x)<<UART0_D_R7T7_SHIFT))&UART0_D_R7T7_MASK)

/* MA1 Bit Fields */

#define UART0_MA1_MA_MASK          0xFFu
#define UART0_MA1_MA_SHIFT         0
#define UART0_MA1_MA_WIDTH         8

#define UART0_MA1_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA1_MA_SHIFT))&UART0_MA1_MA_MASK)

/* MA2 Bit Fields */

#define UART0_MA2_MA_MASK          0xFFu
#define UART0_MA2_MA_SHIFT         0
#define UART0_MA2_MA_WIDTH         8

#define UART0_MA2_MA(x)
(((uint8_t)((uint8_t)(x)<<UART0_MA2_MA_SHIFT))&UART0_MA2_MA_MASK)

/* C4 Bit Fields */

#define UART0_C4_OSR_MASK          0x1Fu
#define UART0_C4_OSR_SHIFT         0
#define UART0_C4_OSR_WIDTH         5

#define UART0_C4_OSR(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_OSR_SHIFT))&UART0_C4_OSR_MASK)

#define UART0_C4_M10_MASK          0x20u
#define UART0_C4_M10_SHIFT         5
#define UART0_C4_M10_WIDTH         1

#define UART0_C4_M10(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_M10_SHIFT))&UART0_C4_M10_MASK)

#define UART0_C4_MAEN2_MASK         0x40u
#define UART0_C4_MAEN2_SHIFT        6
#define UART0_C4_MAEN2_WIDTH        1

#define UART0_C4_MAEN2(x)
(((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN2_SHIFT))&UART0_C4_MAEN2_MASK)

#define UART0_C4_MAEN1_MASK         0x80u

```

```

#define UART0_C4_MAEN1_SHIFT          7
#define UART0_C4_MAEN1_WIDTH          1
#define UART0_C4_MAEN1(x) (((uint8_t)((uint8_t)(x)<<UART0_C4_MAEN1_SHIFT))&UART0_C4_MAEN1_MASK)

/* C5 Bit Fields */

#define UART0_C5_RESETDISC_MASK        0x1u
#define UART0_C5_RESETDISC_SHIFT       0
#define UART0_C5_RESETDISC_WIDTH       1
#define UART0_C5_RESETDISC(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RESETDISC_SHIFT))&UART0_C5_RESETDISC_MASK)

#define UART0_C5_BOTHEDGE_MASK         0x2u
#define UART0_C5_BOTHEDGE_SHIFT        1
#define UART0_C5_BOTHEDGE_WIDTH        1
#define UART0_C5_BOTHEDGE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_BOTHEDGE_SHIFT))&UART0_C5_BOTHEDGE_MASK)

#define UART0_C5_RDMAE_MASK            0x20u
#define UART0_C5_RDMAE_SHIFT           5
#define UART0_C5_RDMAE_WIDTH           1
#define UART0_C5_RDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_RDMAE_SHIFT))&UART0_C5_RDMAE_MASK)

#define UART0_C5_TDMAE_MASK            0x80u
#define UART0_C5_TDMAE_SHIFT           7
#define UART0_C5_TDMAE_WIDTH           1
#define UART0_C5_TDMAE(x) (((uint8_t)((uint8_t)(x)<<UART0_C5_TDMAE_SHIFT))&UART0_C5_TDMAE_MASK)

*/
* @}
/* end of group UART0_Register_Masks */

```

```

/* UART0 - Peripheral instance base addresses */

/** Peripheral UART0 base address */

#define UART0_BASE          (0x4006A000u)

/** Peripheral UART0 base pointer */

#define UART0          ((UART0_Type *)UART0_BASE)

#define UART0_BASE_PTR      (UART0)

/** Array initializer of UART0 peripheral base addresses */

#define UART0_BASE_ADDRS     { UART0_BASE }

/** Array initializer of UART0 peripheral base pointers */

#define UART0_BASE_PTRS      { UART0 }

/*
-----*
-- UART0 - Register accessor macros
----- */
/*!

* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor macros
* @{
*/
/* UART0 - Register instance definitions */

/* UART0 */

#define UART0_BDH          UART0_BDH_REG(UART0)
#define UART0_BDL          UART0_BDL_REG(UART0)
#define UART0_C1           UART0_C1_REG(UART0)
#define UART0_C2           UART0_C2_REG(UART0)
#define UART0_S1           UART0_S1_REG(UART0)

```

```
#define UART0_S2          UART0_S2_REG(UART0)
#define UART0_C3          UART0_C3_REG(UART0)
#define UART0_D           UART0_D_REG(UART0)
#define UART0_MA1         UART0_MA1_REG(UART0)
#define UART0_MA2         UART0_MA2_REG(UART0)
#define UART0_C4           UART0_C4_REG(UART0)
#define UART0_C5           UART0_C5_REG(UART0)
```

```
/*!
 * @}
 */
/* end of group UART0_Register_Accessor_Macros */
```

```
/*!
 * @}
 */
/* end of group UART0_Peripheral_Access_Layer */
```

```
/* -----
-- USB Peripheral Access Layer
----- */
```

```
/*
 * @addtogroup USB_Peripheral_Access_Layer USB Peripheral Access Layer
 * @{
 */

```

```
/** USB - Register Layout Typedef */
```

```
typedef struct {
```

```
__I uint8_t PERID;           /**< Peripheral ID register, offset: 0x0 */

    uint8_t RESERVED_0[3];

__I uint8_t IDCOMP;          /**< Peripheral ID Complement register, offset: 0x4 */

    uint8_t RESERVED_1[3];

__I uint8_t REV;             /**< Peripheral Revision register, offset: 0x8 */

    uint8_t RESERVED_2[3];

__I uint8_t ADDINFO;         /**< Peripheral Additional Info register, offset: 0xC */

    uint8_t RESERVED_3[3];

__IO uint8_t OTGISTAT;       /**< OTG Interrupt Status register, offset: 0x10 */

    uint8_t RESERVED_4[3];

__IO uint8_t OTGICR;          /**< OTG Interrupt Control Register, offset: 0x14 */

    uint8_t RESERVED_5[3];

__IO uint8_t OTGSTAT;         /**< OTG Status register, offset: 0x18 */

    uint8_t RESERVED_6[3];

__IO uint8_t OTGCTL;          /**< OTG Control register, offset: 0x1C */

    uint8_t RESERVED_7[99];

__IO uint8_t ISTAT;           /**< Interrupt Status register, offset: 0x80 */

    uint8_t RESERVED_8[3];

__IO uint8_t INTEN;           /**< Interrupt Enable register, offset: 0x84 */

    uint8_t RESERVED_9[3];

__IO uint8_t ERRSTAT;         /**< Error Interrupt Status register, offset: 0x88 */

    uint8_t RESERVED_10[3];

__IO uint8_t ERREN;            /**< Error Interrupt Enable register, offset: 0x8C */

    uint8_t RESERVED_11[3];

__I uint8_t STAT;              /**< Status register, offset: 0x90 */

    uint8_t RESERVED_12[3];

__IO uint8_t CTL;               /**< Control register, offset: 0x94 */

    uint8_t RESERVED_13[3];

__IO uint8_t ADDR;              /**< Address register, offset: 0x98 */
```

```

    uint8_t RESERVED_14[3];

    __IO uint8_t BDTPAGE1;           /**< BDT Page Register 1, offset: 0x9C */

    uint8_t RESERVED_15[3];

    __IO uint8_t FRMNUML;          /**< Frame Number Register Low, offset: 0xA0 */

    uint8_t RESERVED_16[3];

    __IO uint8_t FRMNUMH;          /**< Frame Number Register High, offset: 0xA4 */

    uint8_t RESERVED_17[3];

    __IO uint8_t TOKEN;            /**< Token register, offset: 0xA8 */

    uint8_t RESERVED_18[3];

    __IO uint8_t SOFTHLD;          /**< SOF Threshold Register, offset: 0xAC */

    uint8_t RESERVED_19[3];

    __IO uint8_t BDTPAGE2;          /**< BDT Page Register 2, offset: 0xB0 */

    uint8_t RESERVED_20[3];

    __IO uint8_t BDTPAGE3;          /**< BDT Page Register 3, offset: 0xB4 */

    uint8_t RESERVED_21[11];

struct {                           /* offset: 0xC0, array step: 0x4 */

    __IO uint8_t ENDPT;           /**< Endpoint Control register, array offset: 0xC0, array step:
                                  0x4 */

    uint8_t RESERVED_0[3];

} ENDPOINT[16];

    __IO uint8_t USBCTRL;          /**< USB Control register, offset: 0x100 */

    uint8_t RESERVED_22[3];

    __I uint8_t OBSERVE;          /**< USB OTG Observe register, offset: 0x104 */

    uint8_t RESERVED_23[3];

    __IO uint8_t CONTROL;          /**< USB OTG Control register, offset: 0x108 */

    uint8_t RESERVED_24[3];

    __IO uint8_t USBTRCO;          /**< USB Transceiver Control Register 0, offset: 0x10C */

    uint8_t RESERVED_25[7];

    __IO uint8_t USBFRMADJUST;     /**< Frame Adjust Register, offset: 0x114 */

```

```

} USB_Type, *USB_MemMapPtr;

/*
-----  

-- USB - Register accessor macros  

----- */
/*!  

* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros  

* @{
* /  

  

/* USB - Register accessors */  

#define USB_PERID_REG(base) ((base)->PERID)  

#define USB_IDCOMP_REG(base) ((base)->IDCOMP)  

#define USB_REV_REG(base) ((base)->REV)  

#define USB_ADDINFO_REG(base) ((base)->ADDINFO)  

#define USB_OTGISTAT_REG(base) ((base)->OTGISTAT)  

#define USB_OTGICR_REG(base) ((base)->OTGICR)  

#define USB_OTGSTAT_REG(base) ((base)->OTGSTAT)  

#define USB_OTGCTL_REG(base) ((base)->OTGCTL)  

#define USB_ISTAT_REG(base) ((base)->ISTAT)  

#define USB_INTEN_REG(base) ((base)->INTEN)  

#define USB_ERRSTAT_REG(base) ((base)->ERRSTAT)  

#define USB_ERREN_REG(base) ((base)->ERREN)  

#define USB_STAT_REG(base) ((base)->STAT)  

#define USB_CTL_REG(base) ((base)->CTL)  

#define USB_ADDR_REG(base) ((base)->ADDR)  

#define USB_BDTPAGE1_REG(base) ((base)->BDTPAGE1)

```

```
#define USB_FRMNUML_REG(base)          ((base)->FRMNUML)
#define USB_FRMNUMH_REG(base)          ((base)->FRMNUMH)
#define USB_TOKEN_REG(base)           ((base)->TOKEN)
#define USB_SOFTHLD_REG(base)          ((base)->SOFTHLD)
#define USB_BDTPAGE2_REG(base)          ((base)->BDTPAGE2)
#define USB_BDTPAGE3_REG(base)          ((base)->BDTPAGE3)
#define USB_ENDPT_REG(base,index)      ((base)->ENDPOINT[index].ENDPT)
#define USB_ENDPT_COUNT                16
#define USB_USBCTRL_REG(base)          ((base)->USBCTRL)
#define USB_OBSERVE_REG(base)          ((base)->OBSERVE)
#define USB_CONTROL_REG(base)          ((base)->CONTROL)
#define USB_USBTRCO_REG(base)          ((base)->USBTRCO)
#define USB_USBFRMADJUST_REG(base)     ((base)->USBFRMADJUST)
```

```
/*!
 * @}
 */
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*
-----  
-- USB Register Masks  
----- */
```

```
/*!
 * @addtogroup USB_Register_Masks USB Register Masks
 * @{
 */

```

```
/* PERID Bit Fields */
```

```

#define USB_PERID_ID_MASK          0x3Fu
#define USB_PERID_ID_SHIFT         0
#define USB_PERID_ID_WIDTH         6
#define USB_PERID_ID(x) (((uint8_t)((uint8_t)(x))<<USB_PERID_ID_SHIFT))&USB_PERID_ID_MASK

/* IDCOMP Bit Fields */

#define USB_IDCOMP_NID_MASK        0x3Fu
#define USB_IDCOMP_NID_SHIFT       0
#define USB_IDCOMP_NID_WIDTH       6
#define USB_IDCOMP_NID(x) (((uint8_t)((uint8_t)(x))<<USB_IDCOMP_NID_SHIFT))&USB_IDCOMP_NID_MASK

/* REV Bit Fields */

#define USB_REV_REV_MASK           0xFFu
#define USB_REV_REV_SHIFT          0
#define USB_REV_REV_WIDTH          8
#define USB_REV_REV(x) (((uint8_t)((uint8_t)(x))<<USB_REV_REV_SHIFT))&USB_REV_REV_MASK

/* ADDINFO Bit Fields */

#define USB_ADDINFO_IHOST_MASK      0x1u
#define USB_ADDINFO_IHOST_SHIFT     0
#define USB_ADDINFO_IHOST_WIDTH     1
#define USB_ADDINFO_IHOST(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IHOST_SHIFT))&USB_ADDINFO_IHOST_MASK

#define USB_ADDINFO_IRQNUM_MASK     0xF8u
#define USB_ADDINFO_IRQNUM_SHIFT    3
#define USB_ADDINFO_IRQNUM_WIDTH    5
#define USB_ADDINFO_IRQNUM(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IRQNUM_SHIFT))&USB_ADDINFO_IRQNUM_MASK

/* OTGISTAT Bit Fields */

#define USB_OTGISTAT_AVBUSCHG_MASK  0x1u

```

```

#define USB_OTGISTAT_AVBUSCHG_SHIFT      0
#define USB_OTGISTAT_AVBUSCHG_WIDTH      1
#define USB_OTGISTAT_AVBUSCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_AVBUSCHG_SHIFT))&USB_OTGISTAT_AVBUSCHG_MASK)

#define USB_OTGISTAT_B_SESS_CHG_MASK      0x4u
#define USB_OTGISTAT_B_SESS_CHG_SHIFT     2
#define USB_OTGISTAT_B_SESS_CHG_WIDTH    1
#define USB_OTGISTAT_B_SESS_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_B_SESS_CHG_SHIFT))&USB_OTGISTAT_B_SESS_CHG_MASK)

#define USB_OTGISTAT_SESSVLDCHG_MASK      0x8u
#define USB_OTGISTAT_SESSVLDCHG_SHIFT     3
#define USB_OTGISTAT_SESSVLDCHG_WIDTH    1
#define USB_OTGISTAT_SESSVLDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_SESSVLDCHG_SHIFT))&USB_OTGISTAT_SESSVLDCHG_MASK)

#define USB_OTGISTAT_LINE_STATE_CHG_MASK   0x20u
#define USB_OTGISTAT_LINE_STATE_CHG_SHIFT  5
#define USB_OTGISTAT_LINE_STATE_CHG_WIDTH  1
#define USB_OTGISTAT_LINE_STATE_CHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_LINE_STATE_CHG_SHIFT))&USB_OTGISTAT_LINE_STATE_CHG_MASK)

#define USB_OTGISTAT_ONEMSEC_MASK         0x40u
#define USB_OTGISTAT_ONEMSEC_SHIFT       6
#define USB_OTGISTAT_ONEMSEC_WIDTH      1
#define USB_OTGISTAT_ONEMSEC(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_ONEMSEC_SHIFT))&USB_OTGISTAT_ONEMSEC_MASK)

#define USB_OTGISTAT_IDCHG_MASK          0x80u
#define USB_OTGISTAT_IDCHG_SHIFT        7
#define USB_OTGISTAT_IDCHG_WIDTH       1
#define USB_OTGISTAT_IDCHG(x)
(((uint8_t)((uint8_t)(x)<<USB_OTGISTAT_IDCHG_SHIFT))&USB_OTGISTAT_IDCHG_MASK)

/* OTGICR Bit Fields */

```

```

#define USB_OTGICR_AVBUSEN_MASK          0x1u
#define USB_OTGICR_AVBUSEN_SHIFT         0
#define USB_OTGICR_AVBUSEN_WIDTH         1
#define USB_OTGICR_AVBUSEN(x)           (((uint8_t)((uint8_t)(x))<<USB_OTGICR_AVBUSEN_SHIFT))&USB_OTGICR_AVBUSEN_MASK)
#define USB_OTGICR_BSESSEN_MASK          0x4u
#define USB_OTGICR_BSESSEN_SHIFT         2
#define USB_OTGICR_BSESSEN_WIDTH         1
#define USB_OTGICR_BSESSEN(x)           (((uint8_t)((uint8_t)(x))<<USB_OTGICR_BSESSEN_SHIFT))&USB_OTGICR_BSESSEN_MASK)
#define USB_OTGICR_SESSVLDEN_MASK        0x8u
#define USB_OTGICR_SESSVLDEN_SHIFT       3
#define USB_OTGICR_SESSVLDEN_WIDTH       1
#define USB_OTGICR_SESSVLDEN(x)         (((uint8_t)((uint8_t)(x))<<USB_OTGICR_SESSVLDEN_SHIFT))&USB_OTGICR_SESSVLDEN_MASK)
#define USB_OTGICR_LINESTATEEN_MASK      0x20u
#define USB_OTGICR_LINESTATEEN_SHIFT     5
#define USB_OTGICR_LINESTATEEN_WIDTH     1
#define USB_OTGICR_LINESTATEEN(x)        (((uint8_t)((uint8_t)(x))<<USB_OTGICR_LINESTATEEN_SHIFT))&USB_OTGICR_LINESTATEEN_MASK)
#define USB_OTGICR_ONEMSECEN_MASK        0x40u
#define USB_OTGICR_ONEMSECEN_SHIFT       6
#define USB_OTGICR_ONEMSECEN_WIDTH       1
#define USB_OTGICR_ONEMSECEN(x)         (((uint8_t)((uint8_t)(x))<<USB_OTGICR_ONEMSECEN_SHIFT))&USB_OTGICR_ONEMSECEN_MASK)
#define USB_OTGICR_IDEN_MASK             0x80u
#define USB_OTGICR_IDEN_SHIFT            7
#define USB_OTGICR_IDEN_WIDTH            1
#define USB_OTGICR_IDEN(x)              (((uint8_t)((uint8_t)(x))<<USB_OTGICR_IDEN_SHIFT))&USB_OTGICR_IDEN_MASK)
/* OTGSTAT Bit Fields */

```

```

#define USB_OTGSTAT_AVBUSVLD_MASK      0x1u
#define USB_OTGSTAT_AVBUSVLD_SHIFT     0
#define USB_OTGSTAT_AVBUSVLD_WIDTH     1
#define USB_OTGSTAT_AVBUSVLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_AVBUSVLD_SHIFT))&USB_OTGSTAT_AVBUSVLD_MASK

#define USB_OTGSTAT_BSESSEND_MASK       0x4u
#define USB_OTGSTAT_BSESSEND_SHIFT     2
#define USB_OTGSTAT_BSESSEND_WIDTH     1
#define USB_OTGSTAT_BSESSEND(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_BSESSEND_SHIFT))&USB_OTGSTAT_BSESSEND_MASK

#define USB_OTGSTAT_SESS_VLD_MASK       0x8u
#define USB_OTGSTAT_SESS_VLD_SHIFT     3
#define USB_OTGSTAT_SESS_VLD_WIDTH     1
#define USB_OTGSTAT_SESS_VLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_SESS_VLD_SHIFT))&USB_OTGSTAT_SESS_VLD_MASK

#define USB_OTGSTAT_LINESTATESTABLE_MASK 0x20u
#define USB_OTGSTAT_LINESTATESTABLE_SHIFT 5
#define USB_OTGSTAT_LINESTATESTABLE_WIDTH 1
#define USB_OTGSTAT_LINESTATESTABLE(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_LINESTATESTABLE_SHIFT))&USB_OTGSTAT_LINESTATESTABLE_MASK

#define USB_OTGSTAT_ONEMSECEN_MASK      0x40u
#define USB_OTGSTAT_ONEMSECEN_SHIFT     6
#define USB_OTGSTAT_ONEMSECEN_WIDTH     1
#define USB_OTGSTAT_ONEMSECEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ONEMSECEN_SHIFT))&USB_OTGSTAT_ONEMSECEN_MASK

#define USB_OTGSTAT_ID_MASK            0x80u
#define USB_OTGSTAT_ID_SHIFT          7
#define USB_OTGSTAT_ID_WIDTH          1
#define USB_OTGSTAT_ID(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ID_SHIFT))&USB_OTGSTAT_ID_MASK

```

```

/* OTGCTL Bit Fields */

#define USB_OTGCTL_OTGEN_MASK          0x4u
#define USB_OTGCTL_OTGEN_SHIFT         2
#define USB_OTGCTL_OTGEN_WIDTH         1
#define USB_OTGCTL_OTGEN(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_OTGEN_SHIFT))&USB_OTGCTL_OTGEN_MASK)

#define USB_OTGCTL_DMLOW_MASK          0x10u
#define USB_OTGCTL_DMLOW_SHIFT         4
#define USB_OTGCTL_DMLOW_WIDTH         1
#define USB_OTGCTL_DMLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DMLOW_SHIFT))&USB_OTGCTL_DMLOW_MASK)

#define USB_OTGCTL_DPLOW_MASK          0x20u
#define USB_OTGCTL_DPLOW_SHIFT         5
#define USB_OTGCTL_DPLOW_WIDTH         1
#define USB_OTGCTL_DPLOW(x)            (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPLOW_SHIFT))&USB_OTGCTL_DPLOW_MASK)

#define USB_OTGCTL_DPHIGH_MASK         0x80u
#define USB_OTGCTL_DPHIGH_SHIFT        7
#define USB_OTGCTL_DPHIGH_WIDTH        1
#define USB_OTGCTL_DPHIGH(x)           (((uint8_t)((uint8_t)(x)<<USB_OTGCTL_DPHIGH_SHIFT))&USB_OTGCTL_DPHIGH_MASK)

/* ISTAT Bit Fields */

#define USB_ISTAT_USBRST_MASK          0x1u
#define USB_ISTAT_USBRST_SHIFT         0
#define USB_ISTAT_USBRST_WIDTH         1
#define USB_ISTAT_USBRST(x)            (((uint8_t)((uint8_t)(x)<<USB_ISTAT_USBRST_SHIFT))&USB_ISTAT_USBRST_MASK)

#define USB_ISTAT_ERROR_MASK           0x2u
#define USB_ISTAT_ERROR_SHIFT          1
#define USB_ISTAT_ERROR_WIDTH          1

```

```

#define USB_ISTAT_ERROR(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ERROR_SHIFT))&USB_ISTAT_ERROR_MASK)

#define USB_ISTAT_SOFTOK_MASK          0x4u
#define USB_ISTAT_SOFTOK_SHIFT        2
#define USB_ISTAT_SOFTOK_WIDTH        1

#define USB_ISTAT_SOFTOK(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SOFTOK_SHIFT))&USB_ISTAT_SOFTOK_MASK)

#define USB_ISTAT_TOKDNE_MASK         0x8u
#define USB_ISTAT_TOKDNE_SHIFT        3
#define USB_ISTAT_TOKDNE_WIDTH        1

#define USB_ISTAT_TOKDNE(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_TOKDNE_SHIFT))&USB_ISTAT_TOKDNE_MASK)

#define USB_ISTAT_SLEEP_MASK          0x10u
#define USB_ISTAT_SLEEP_SHIFT         4
#define USB_ISTAT_SLEEP_WIDTH         1

#define USB_ISTAT_SLEEP(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_SLEEP_SHIFT))&USB_ISTAT_SLEEP_MASK)

#define USB_ISTAT_RESUME_MASK         0x20u
#define USB_ISTAT_RESUME_SHIFT        5
#define USB_ISTAT_RESUME_WIDTH        1

#define USB_ISTAT_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_RESUME_SHIFT))&USB_ISTAT_RESUME_MASK)

#define USB_ISTAT_ATTACH_MASK          0x40u
#define USB_ISTAT_ATTACH_SHIFT         6
#define USB_ISTAT_ATTACH_WIDTH         1

#define USB_ISTAT_ATTACH(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_ATTACH_SHIFT))&USB_ISTAT_ATTACH_MASK)

#define USB_ISTAT_STALL_MASK          0x80u
#define USB_ISTAT_STALL_SHIFT         7
#define USB_ISTAT_STALL_WIDTH         1

```

```

#define USB_ISTAT_STALL(x)
(((uint8_t)((uint8_t)(x)<<USB_ISTAT_STALL_SHIFT))&USB_ISTAT_STALL_MASK)

/* INTEN Bit Fields */

#define USB_INTEN_USBRSTEN_MASK          0x1u
#define USB_INTEN_USBRSTEN_SHIFT         0
#define USB_INTEN_USBRSTEN_WIDTH         1
#define USB_INTEN_USBRSTEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_USBRSTEN_SHIFT))&USB_INTEN_USBRSTEN_MASK)

#define USB_INTEN_ERROREN_MASK           0x2u
#define USB_INTEN_ERROREN_SHIFT          1
#define USB_INTEN_ERROREN_WIDTH          1
#define USB_INTEN_ERROREN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ERROREN_SHIFT))&USB_INTEN_ERROREN_MASK)

#define USB_INTEN_SOFTOKEN_MASK          0x4u
#define USB_INTEN_SOFTOKEN_SHIFT         2
#define USB_INTEN_SOFTOKEN_WIDTH         1
#define USB_INTEN_SOFTOKEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SOFTOKEN_SHIFT))&USB_INTEN_SOFTOKEN_MASK)

#define USB_INTEN_TOKDNEEN_MASK          0x8u
#define USB_INTEN_TOKDNEEN_SHIFT         3
#define USB_INTEN_TOKDNEEN_WIDTH         1
#define USB_INTEN_TOKDNEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_TOKDNEEN_SHIFT))&USB_INTEN_TOKDNEEN_MASK)

#define USB_INTEN_SLEEPEN_MASK           0x10u
#define USB_INTEN_SLEEPEN_SHIFT          4
#define USB_INTEN_SLEEPEN_WIDTH          1
#define USB_INTEN_SLEEPEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_SLEEPEN_SHIFT))&USB_INTEN_SLEEPEN_MASK)

#define USB_INTEN_RESUMEEN_MASK          0x20u
#define USB_INTEN_RESUMEEN_SHIFT         5
#define USB_INTEN_RESUMEEN_WIDTH         1

```

```

#define USB_INTEN_RESUMEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_RESUMEEN_SHIFT))&USB_INTEN_RESUMEEN_MASK)

#define USB_INTEN_ATTACHEN_MASK          0x40u
#define USB_INTEN_ATTACHEN_SHIFT        6
#define USB_INTEN_ATTACHEN_WIDTH        1

#define USB_INTEN_ATTACHEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_ATTACHEN_SHIFT))&USB_INTEN_ATTACHEN_MASK)

#define USB_INTEN_STALLEN_MASK          0x80u
#define USB_INTEN_STALLEN_SHIFT        7
#define USB_INTEN_STALLEN_WIDTH        1

#define USB_INTEN_STALLEN(x)
(((uint8_t)((uint8_t)(x)<<USB_INTEN_STALLEN_SHIFT))&USB_INTEN_STALLEN_MASK)

/* ERRSTAT Bit Fields */

#define USB_ERRSTAT_PIDERR_MASK         0x1u
#define USB_ERRSTAT_PIDERR_SHIFT       0
#define USB_ERRSTAT_PIDERR_WIDTH       1

#define USB_ERRSTAT_PIDERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_PIDERR_SHIFT))&USB_ERRSTAT_PIDERR_MASK)

#define USB_ERRSTAT_CRC5EOF_MASK         0x2u
#define USB_ERRSTAT_CRC5EOF_SHIFT       1
#define USB_ERRSTAT_CRC5EOF_WIDTH       1

#define USB_ERRSTAT_CRC5EOF(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC5EOF_SHIFT))&USB_ERRSTAT_CRC5EOF_MASK)

#define USB_ERRSTAT_CRC16_MASK          0x4u
#define USB_ERRSTAT_CRC16_SHIFT        2
#define USB_ERRSTAT_CRC16_WIDTH        1

#define USB_ERRSTAT_CRC16(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_CRC16_SHIFT))&USB_ERRSTAT_CRC16_MASK)

#define USB_ERRSTAT_DFN8_MASK           0x8u
#define USB_ERRSTAT_DFN8_SHIFT         3
#define USB_ERRSTAT_DFN8_WIDTH         1

```

```

#define USB_ERRSTAT_DFN8(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DFN8_SHIFT))&USB_ERRSTAT_DFN8_MASK)

#define USB_ERRSTAT_BTOERR_MASK          0x10u
#define USB_ERRSTAT_BTOERR_SHIFT        4
#define USB_ERRSTAT_BTOERR_WIDTH        1

#define USB_ERRSTAT_BTOERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTOERR_SHIFT))&USB_ERRSTAT_BTOERR_MASK)

#define USB_ERRSTAT_DMAERR_MASK         0x20u
#define USB_ERRSTAT_DMAERR_SHIFT       5
#define USB_ERRSTAT_DMAERR_WIDTH       1

#define USB_ERRSTAT_DMAERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_DMAERR_SHIFT))&USB_ERRSTAT_DMAERR_MASK)

#define USB_ERRSTAT_BTSERR_MASK         0x80u
#define USB_ERRSTAT_BTSERR_SHIFT       7
#define USB_ERRSTAT_BTSERR_WIDTH       1

#define USB_ERRSTAT_BTSERR(x)
(((uint8_t)((uint8_t)(x)<<USB_ERRSTAT_BTSERR_SHIFT))&USB_ERRSTAT_BTSERR_MASK)

/* ERREN Bit Fields */

#define USB_ERREN_PIDERREN_MASK         0x1u
#define USB_ERREN_PIDERREN_SHIFT        0
#define USB_ERREN_PIDERREN_WIDTH        1

#define USB_ERREN_PIDERREN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_PIDERREN_SHIFT))&USB_ERREN_PIDERREN_MASK)

#define USB_ERREN_CRC5EOFEN_MASK        0x2u
#define USB_ERREN_CRC5EOFEN_SHIFT      1
#define USB_ERREN_CRC5EOFEN_WIDTH      1

#define USB_ERREN_CRC5EOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ERREN_CRC5EOFEN_SHIFT))&USB_ERREN_CRC5EOFEN_MASK)

#define USB_ERREN_CRC16EN_MASK          0x4u
#define USB_ERREN_CRC16EN_SHIFT        2
#define USB_ERREN_CRC16EN_WIDTH        1

```

```

#define USB_ERREN_CRC16EN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_CRC16EN_SHIFT))&USB_ERREN_CRC16EN_MASK)

#define USB_ERREN_DFN8EN_MASK          0x8u
#define USB_ERREN_DFN8EN_SHIFT         3
#define USB_ERREN_DFN8EN_WIDTH         1

#define USB_ERREN_DFN8EN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_DFN8EN_SHIFT))&USB_ERREN_DFN8EN_MASK)

#define USB_ERREN_BTOERREN_MASK        0x10u
#define USB_ERREN_BTOERREN_SHIFT       4
#define USB_ERREN_BTOERREN_WIDTH       1

#define USB_ERREN_BTOERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_BTOERREN_SHIFT))&USB_ERREN_BTOERREN_MASK)

#define USB_ERREN_DMAERREN_MASK        0x20u
#define USB_ERREN_DMAERREN_SHIFT       5
#define USB_ERREN_DMAERREN_WIDTH       1

#define USB_ERREN_DMAERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_DMAERREN_SHIFT))&USB_ERREN_DMAERREN_MASK)

#define USB_ERREN_BTSERREN_MASK        0x80u
#define USB_ERREN_BTSERREN_SHIFT       7
#define USB_ERREN_BTSERREN_WIDTH       1

#define USB_ERREN_BTSERREN(x)
(((uint8_t)((uint8_t)(x))<<USB_ERREN_BTSERREN_SHIFT))&USB_ERREN_BTSERREN_MASK)

/* STAT Bit Fields */

#define USB_STAT_ODD_MASK             0x4u
#define USB_STAT_ODD_SHIFT            2
#define USB_STAT_ODD_WIDTH            1

#define USB_STAT_ODD(x)
(((uint8_t)((uint8_t)(x))<<USB_STAT_ODD_SHIFT))&USB_STAT_ODD_MASK)

#define USB_STAT_TX_MASK              0x8u
#define USB_STAT_TX_SHIFT             3
#define USB_STAT_TX_WIDTH             1

```

```

#define USB_STAT_TX(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_TX_SHIFT))&USB_STAT_TX_MASK)

#define USB_STAT_ENDP_MASK          0xF0u
#define USB_STAT_ENDP_SHIFT        4
#define USB_STAT_ENDP_WIDTH        4

#define USB_STAT_ENDP(x)
(((uint8_t)((uint8_t)(x)<<USB_STAT_ENDP_SHIFT))&USB_STAT_ENDP_MASK)

/* CTL Bit Fields */

#define USB_CTL_USBENSOFEN_MASK      0x1u
#define USB_CTL_USBENSOFEN_SHIFT    0
#define USB_CTL_USBENSOFEN_WIDTH    1

#define USB_CTL_USBENSOFEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_USBENSOFEN_SHIFT))&USB_CTL_USBENSOFEN_MASK)

#define USB_CTL_ODDRST_MASK         0x2u
#define USB_CTL_ODDRST_SHIFT       1
#define USB_CTL_ODDRST_WIDTH       1

#define USB_CTL_ODDRST(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_ODDRST_SHIFT))&USB_CTL_ODDRST_MASK)

#define USB_CTL_RESUME_MASK         0x4u
#define USB_CTL_RESUME_SHIFT       2
#define USB_CTL_RESUME_WIDTH       1

#define USB_CTL_RESUME(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESUME_SHIFT))&USB_CTL_RESUME_MASK)

#define USB_CTL_HOSTMODEEN_MASK     0x8u
#define USB_CTL_HOSTMODEEN_SHIFT   3
#define USB_CTL_HOSTMODEEN_WIDTH   1

#define USB_CTL_HOSTMODEEN(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_HOSTMODEEN_SHIFT))&USB_CTL_HOSTMODEEN_MASK)

#define USB_CTL_RESET_MASK          0x10u
#define USB_CTL_RESET_SHIFT        4
#define USB_CTL_RESET_WIDTH        1

```

```

#define USB_CTL_RESET(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_RESET_SHIFT))&USB_CTL_RESET_MASK)

#define USB_CTL_TXSUSPENDTOKENBUSY_MASK      0x20u
#define USB_CTL_TXSUSPENDTOKENBUSY_SHIFT     5
#define USB_CTL_TXSUSPENDTOKENBUSY_WIDTH     1

#define USB_CTL_TXSUSPENDTOKENBUSY(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_TXSUSPENDTOKENBUSY_SHIFT))&USB_CTL_TXSUSPENDTOKENBUSY
 _MASK)

#define USB_CTL_SE0_MASK                    0x40u
#define USB_CTL_SE0_SHIFT                  6
#define USB_CTL_SE0_WIDTH                 1

#define USB_CTL_SE0(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_SE0_SHIFT))&USB_CTL_SE0_MASK)

#define USB_CTL_JSTATE_MASK                0x80u
#define USB_CTL_JSTATE_SHIFT              7
#define USB_CTL_JSTATE_WIDTH              1

#define USB_CTL_JSTATE(x)
(((uint8_t)((uint8_t)(x)<<USB_CTL_JSTATE_SHIFT))&USB_CTL_JSTATE_MASK)

/* ADDR Bit Fields */

#define USB_ADDR_ADDR_MASK                0x7Fu
#define USB_ADDR_ADDR_SHIFT              0
#define USB_ADDR_ADDR_WIDTH              7

#define USB_ADDR_ADDR(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_ADDR_SHIFT))&USB_ADDR_ADDR_MASK)

#define USB_ADDR_LSEN_MASK                0x80u
#define USB_ADDR_LSEN_SHIFT              7
#define USB_ADDR_LSEN_WIDTH              1

#define USB_ADDR_LSEN(x)
(((uint8_t)((uint8_t)(x)<<USB_ADDR_LSEN_SHIFT))&USB_ADDR_LSEN_MASK)

/* BDTPAGE1 Bit Fields */

#define USB_BDTPAGE1_BDTBA_MASK          0xFEu

```

```

#define USB_BDTPAGE1_BDTBA_SHIFT      1
#define USB_BDTPAGE1_BDTBA_WIDTH      7
#define USB_BDTPAGE1_BDTBA(x)
(((uint8_t)((uint8_t)(x)<<USB_BDTPAGE1_BDTBA_SHIFT))&USB_BDTPAGE1_BDTBA_MASK)

/* FRMNUML Bit Fields */

#define USB_FRMNUML_FRM_MASK          0xFFu
#define USB_FRMNUML_FRM_SHIFT         0
#define USB_FRMNUML_FRM_WIDTH         8
#define USB_FRMNUML_FRM(x)
(((uint8_t)((uint8_t)(x)<<USB_FRMNUML_FRM_SHIFT))&USB_FRMNUML_FRM_MASK)

/* FRMNUMH Bit Fields */

#define USB_FRMNUMH_FRM_MASK          0x7u
#define USB_FRMNUMH_FRM_SHIFT         0
#define USB_FRMNUMH_FRM_WIDTH         3
#define USB_FRMNUMH_FRM(x)
(((uint8_t)((uint8_t)(x)<<USB_FRMNUMH_FRM_SHIFT))&USB_FRMNUMH_FRM_MASK)

/* TOKEN Bit Fields */

#define USB_TOKEN_TOKENENDPT_MASK      0xFu
#define USB_TOKEN_TOKENENDPT_SHIFT     0
#define USB_TOKEN_TOKENENDPT_WIDTH     4
#define USB_TOKEN_TOKENENDPT(x)
(((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENENDPT_SHIFT))&USB_TOKEN_TOKENENDPT_MASK)

#define USB_TOKEN_TOKENPID_MASK         0xF0u
#define USB_TOKEN_TOKENPID_SHIFT        4
#define USB_TOKEN_TOKENPID_WIDTH        4
#define USB_TOKEN_TOKENPID(x)
(((uint8_t)((uint8_t)(x)<<USB_TOKEN_TOKENPID_SHIFT))&USB_TOKEN_TOKENPID_MASK)

/* SOFTHLD Bit Fields */

#define USB_SOFTHLD_CNT_MASK           0xFFu
#define USB_SOFTHLD_CNT_SHIFT          0

```

```

#define USB_SOFTHLD_CNT_WIDTH          8
#define USB_SOFTHLD_CNT(x) (((uint8_t)((uint8_t)(x)<<USB_SOFTHLD_CNT_SHIFT))&USB_SOFTHLD_CNT_MASK)

/* BDTPAGE2 Bit Fields */

#define USB_BDTPAGE2_BDTBA_MASK        0xFFu
#define USB_BDTPAGE2_BDTBA_SHIFT       0
#define USB_BDTPAGE2_BDTBA_WIDTH       8
#define USB_BDTPAGE2_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE2_BDTBA_SHIFT))&USB_BDTPAGE2_BDTBA_MASK)

/* BDTPAGE3 Bit Fields */

#define USB_BDTPAGE3_BDTBA_MASK        0xFFu
#define USB_BDTPAGE3_BDTBA_SHIFT       0
#define USB_BDTPAGE3_BDTBA_WIDTH       8
#define USB_BDTPAGE3_BDTBA(x) (((uint8_t)((uint8_t)(x)<<USB_BDTPAGE3_BDTBA_SHIFT))&USB_BDTPAGE3_BDTBA_MASK)

/* ENDPT Bit Fields */

#define USB_ENDPT_EPHSHK_MASK          0x1u
#define USB_ENDPT_EPHSHK_SHIFT         0
#define USB_ENDPT_EPHSHK_WIDTH         1
#define USB_ENDPT_EPHSHK(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPHSHK_SHIFT))&USB_ENDPT_EPHSHK_MASK)

#define USB_ENDPT_EPSTALL_MASK         0x2u
#define USB_ENDPT_EPSTALL_SHIFT        1
#define USB_ENDPT_EPSTALL_WIDTH        1
#define USB_ENDPT_EPSTALL(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPSTALL_SHIFT))&USB_ENDPT_EPSTALL_MASK)

#define USB_ENDPT_EPTXEN_MASK          0x4u
#define USB_ENDPT_EPTXEN_SHIFT         2
#define USB_ENDPT_EPTXEN_WIDTH         1
#define USB_ENDPT_EPTXEN(x) (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPTXEN_SHIFT))&USB_ENDPT_EPTXEN_MASK)

```

```

#define USB_ENDPT_EPRXEN_MASK          0x8u
#define USB_ENDPT_EPRXEN_SHIFT         3
#define USB_ENDPT_EPRXEN_WIDTH         1
#define USB_ENDPT_EPRXEN(x)            (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPRXEN_SHIFT))&USB_ENDPT_EPRXEN_MASK)

#define USB_ENDPT_EPCTLDIS_MASK        0x10u
#define USB_ENDPT_EPCTLDIS_SHIFT       4
#define USB_ENDPT_EPCTLDIS_WIDTH       1
#define USB_ENDPT_EPCTLDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_EPCTLDIS_SHIFT))&USB_ENDPT_EPCTLDIS_MASK)

#define USB_ENDPT_RETRYDIS_MASK        0x40u
#define USB_ENDPT_RETRYDIS_SHIFT       6
#define USB_ENDPT_RETRYDIS_WIDTH       1
#define USB_ENDPT_RETRYDIS(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_RETRYDIS_SHIFT))&USB_ENDPT_RETRYDIS_MASK)

#define USB_ENDPT_HOSTWOHUB_MASK        0x80u
#define USB_ENDPT_HOSTWOHUB_SHIFT       7
#define USB_ENDPT_HOSTWOHUB_WIDTH       1
#define USB_ENDPT_HOSTWOHUB(x)          (((uint8_t)((uint8_t)(x)<<USB_ENDPT_HOSTWOHUB_SHIFT))&USB_ENDPT_HOSTWOHUB_MASK)

/* USBCTRL Bit Fields */

#define USB_USBCTRL_PDE_MASK           0x40u
#define USB_USBCTRL_PDE_SHIFT          6
#define USB_USBCTRL_PDE_WIDTH          1
#define USB_USBCTRL_PDE(x)             (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_PDE_SHIFT))&USB_USBCTRL_PDE_MASK)

#define USB_USBCTRL_SUSP_MASK           0x80u
#define USB_USBCTRL_SUSP_SHIFT          7
#define USB_USBCTRL_SUSP_WIDTH          1
#define USB_USBCTRL_SUSP(x)             (((uint8_t)((uint8_t)(x)<<USB_USBCTRL_SUSP_SHIFT))&USB_USBCTRL_SUSP_MASK)

```

```

/* OBSERVE Bit Fields */

#define USB_OBSERVE_DMPD_MASK          0x10u
#define USB_OBSERVE_DMPD_SHIFT         4
#define USB_OBSERVE_DMPD_WIDTH         1
#define USB_OBSERVE_DMPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DMPD_SHIFT))&USB_OBSERVE_DMPD_MASK)

#define USB_OBSERVE_DPPD_MASK          0x40u
#define USB_OBSERVE_DPPD_SHIFT         6
#define USB_OBSERVE_DPPD_WIDTH         1
#define USB_OBSERVE_DPPD(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPD_SHIFT))&USB_OBSERVE_DPPD_MASK)

#define USB_OBSERVE_DPPU_MASK          0x80u
#define USB_OBSERVE_DPPU_SHIFT         7
#define USB_OBSERVE_DPPU_WIDTH         1
#define USB_OBSERVE_DPPU(x)            (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPU_SHIFT))&USB_OBSERVE_DPPU_MASK)

/* CONTROL Bit Fields */

#define USB_CONTROL_DPPULLUPNONOTG_MASK 0x10u
#define USB_CONTROL_DPPULLUPNONOTG_SHIFT 4
#define USB_CONTROL_DPPULLUPNONOTG_WIDTH 1
#define USB_CONTROL_DPPULLUPNONOTG(x)   (((uint8_t)((uint8_t)(x))<<USB_CONTROL_DPPULLUPNONOTG_SHIFT))&USB_CONTROL_DPPULLUPNONOTG_MASK)

/* USBTRCO Bit Fields */

#define USB_USBTRCO_USB_RESUME_INT_MASK 0x1u
#define USB_USBTRCO_USB_RESUME_INT_SHIFT 0
#define USB_USBTRCO_USB_RESUME_INT_WIDTH 1
#define USB_USBTRCO_USB_RESUME_INT(x)   (((uint8_t)((uint8_t)(x))<<USB_USBTRCO_USB_RESUME_INT_SHIFT))&USB_USBTRCO_USB_RESUME_INT_MASK)

#define USB_USBTRCO_SYNC_DET_MASK      0x2u

```

```

#define USB_USBTRCO_SYNC_DET_SHIFT      1
#define USB_USBTRCO_SYNC_DET_WIDTH      1
#define USB_USBTRCO_SYNC_DET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_SYNC_DET_SHIFT))&USB_USBTRCO_SYNC_DET_MASK)

#define USB_USBTRCO_USBRESMEN_MASK      0x20u
#define USB_USBTRCO_USBRESMEN_SHIFT      5
#define USB_USBTRCO_USBRESMEN_WIDTH      1
#define USB_USBTRCO_USBRESMEN(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESMEN_SHIFT))&USB_USBTRCO_USBRESMEN_MASK)

#define USB_USBTRCO_USBRESET_MASK        0x80u
#define USB_USBTRCO_USBRESET_SHIFT       7
#define USB_USBTRCO_USBRESET_WIDTH       1
#define USB_USBTRCO_USBRESET(x)
(((uint8_t)((uint8_t)(x)<<USB_USBTRCO_USBRESET_SHIFT))&USB_USBTRCO_USBRESET_MASK)

/* USBFRMADJUST Bit Fields */

#define USB_USBFRMADJUST_ADJ_MASK        0xFFu
#define USB_USBFRMADJUST_ADJ_SHIFT       0
#define USB_USBFRMADJUST_ADJ_WIDTH       8
#define USB_USBFRMADJUST_ADJ(x)
(((uint8_t)((uint8_t)(x)<<USB_USBFRMADJUST_ADJ_SHIFT))&USB_USBFRMADJUST_ADJ_MASK)

/*
 * @}
 */

/* end of group USB_Register_Masks */

```

```

/* USB - Peripheral instance base addresses */
/** Peripheral USBO base address */
#define USBO_BASE                      (0x40072000u)
/** Peripheral USBO base pointer */

```

```

#define USBO ((USB_Type *)USBO_BASE)

#define USBO_BASE_PTR (USBO)

/** Array initializer of USB peripheral base addresses */

#define USB_BASE_ADDRS { USBO_BASE }

/** Array initializer of USB peripheral base pointers */

#define USB_BASE_PTRS { USBO }

/*
-- USB - Register accessor macros
*/
/*!
* @addtogroup USB_Register_Accessor_Macros USB - Register accessor macros
* @{
*/

```

```

/* USB - Register instance definitions */

/* USBO */

#define USBO_PERID USB_PERID_REG(USBO)
#define USBO_IDCOMP USB_IDCOMP_REG(USBO)
#define USBO_REV USB_REV_REG(USBO)
#define USBO_ADDINFO USB_ADDINFO_REG(USBO)
#define USBO_OTGISTAT USB_OTGISTAT_REG(USBO)
#define USBO_OTGICR USB_OTGICR_REG(USBO)
#define USBO_OTGSTAT USB_OTGSTAT_REG(USBO)
#define USBO_OTGCTL USB_OTGCTL_REG(USBO)
#define USBO_ISTAT USB_ISTAT_REG(USBO)
#define USBO_INTEN USB_INTEN_REG(USBO)

```

#define USBO_ERRSTAT	USB_ERRSTAT_REG(USBO)
#define USBO_ERREN	USB_ERREN_REG(USBO)
#define USBO_STAT	USB_STAT_REG(USBO)
#define USBO_CTL	USB_CTL_REG(USBO)
#define USBO_ADDR	USB_ADDR_REG(USBO)
#define USBO_BDTPAGE1	USB_BDTPAGE1_REG(USBO)
#define USBO_FRMNUML	USB_FRMNUML_REG(USBO)
#define USBO_FRMNUMH	USB_FRMNUMH_REG(USBO)
#define USBO_TOKEN	USB_TOKEN_REG(USBO)
#define USBO_SOFTHLD	USB_SOFTHLD_REG(USBO)
#define USBO_BDTPAGE2	USB_BDTPAGE2_REG(USBO)
#define USBO_BDTPAGE3	USB_BDTPAGE3_REG(USBO)
#define USBO_ENDPT0	USB_ENDPT_REG(USBO,0)
#define USBO_ENDPT1	USB_ENDPT_REG(USBO,1)
#define USBO_ENDPT2	USB_ENDPT_REG(USBO,2)
#define USBO_ENDPT3	USB_ENDPT_REG(USBO,3)
#define USBO_ENDPT4	USB_ENDPT_REG(USBO,4)
#define USBO_ENDPT5	USB_ENDPT_REG(USBO,5)
#define USBO_ENDPT6	USB_ENDPT_REG(USBO,6)
#define USBO_ENDPT7	USB_ENDPT_REG(USBO,7)
#define USBO_ENDPT8	USB_ENDPT_REG(USBO,8)
#define USBO_ENDPT9	USB_ENDPT_REG(USBO,9)
#define USBO_ENDPT10	USB_ENDPT_REG(USBO,10)
#define USBO_ENDPT11	USB_ENDPT_REG(USBO,11)
#define USBO_ENDPT12	USB_ENDPT_REG(USBO,12)
#define USBO_ENDPT13	USB_ENDPT_REG(USBO,13)
#define USBO_ENDPT14	USB_ENDPT_REG(USBO,14)
#define USBO_ENDPT15	USB_ENDPT_REG(USBO,15)
#define USBO_USBCTRL	USB_USBCTRL_REG(USBO)

```
#define USBO_OBSERVE           USB_OBSERVE_REG(USBO)
#define USBO_CONTROL            USB_CONTROL_REG(USBO)
#define USBO_USBTRC0            USB_USBTRC0_REG(USBO)
#define USBO_USBFRMADJUST       USB_USBFRMADJUST_REG(USBO)
```

```
/* USB - Register array accessors */
```

```
#define USBO_ENDPT(index)      USB_ENDPT_REG(USBO,index)
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Register_Accessor_Macros */
```

```
/*!
```

```
* @}
```

```
/* /* end of group USB_Peripheral_Access_Layer */
```

```
/*
```

```
** End of section using anonymous unions
```

```
*/
```

```
#if defined(__ARMCC_VERSION)
    #pragma pop
#elif defined(__CWCC__)
    #pragma pop
#elif defined(__GNUC__)
    /* leave anonymous unions enabled */
#elif defined(__IAR_SYSTEMS_ICC__)
```

```

#pragma language=default

#else
#error Not supported compiler type

#endif

/*!
 * @}
 */

/* /* end of group Peripheral_access_layer */

/*
-----  

-- Backward Compatibility  

----- */

```

```

/*!  

 * @addtogroup Backward_Compatibility_Symbols Backward Compatibility  

 * @{
 * /  


```

#define DMA_REQC_ARR_DMAC_MASK	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC_SHIFT	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC(x)	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_MASK	This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_SHIFT	This_symbol_has_been_deprecated
#define DMA_REQC0	This_symbol_has_been_deprecated
#define DMA_REQC1	This_symbol_has_been_deprecated
#define DMA_REQC2	This_symbol_has_been_deprecated
#define DMA_REQC3	This_symbol_has_been_deprecated
#define MCG_S_LOLS_MASK	MCG_S_LOLS0_MASK

#define MCG_S_LOLS_SHIFT	MCG_S_LOLS0_SHIFT
#define SIM_FCFG2_MAXADDR_MASK	SIM_FCFG2_MAXADDR0_MASK
#define SIM_FCFG2_MAXADDR_SHIFT	SIM_FCFG2_MAXADDR0_SHIFT
#define SIM_FCFG2_MAXADDR	SIM_FCFG2_MAXADDR0
#define SPI_C2_SPLPIE_MASK	This_symbol_has_been_deprecated
#define SPI_C2_SPLPIE_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_SHIFT	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_MASK	This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_SHIFT	This_symbol_has_been_deprecated
#define UARTLP_Type	UART0_Type
#define UARTLP_BDH_REG	UART0_BDH_REG
#define UARTLP_BDL_REG	UART0_BDL_REG
#define UARTLP_C1_REG	UART0_C1_REG
#define UARTLP_C2_REG	UART0_C2_REG
#define UARTLP_S1_REG	UART0_S1_REG
#define UARTLP_S2_REG	UART0_S2_REG
#define UARTLP_C3_REG	UART0_C3_REG
#define UARTLP_D_REG	UART0_D_REG
#define UARTLP_MA1_REG	UART0_MA1_REG
#define UARTLP_MA2_REG	UART0_MA2_REG
#define UARTLP_C4_REG	UART0_C4_REG
#define UARTLP_C5_REG	UART0_C5_REG
#define UARTLP_BDH_SBR_MASK	UART0_BDH_SBR_MASK
#define UARTLP_BDH_SBR_SHIFT	UART0_BDH_SBR_SHIFT
#define UARTLP_BDH_SBR(x)	UART0_BDH_SBR(x)
#define UARTLP_BDH_SBNS_MASK	UART0_BDH_SBNS_MASK

#define UARLTP_BDH_SBNS_SHIFT	UART0_BDH_SBNS_SHIFT
#define UARLTP_BDH_RXEDGIE_MASK	UART0_BDH_RXEDGIE_MASK
#define UARLTP_BDH_RXEDGIE_SHIFT	UART0_BDH_RXEDGIE_SHIFT
#define UARLTP_BDH_LBKDIE_MASK	UART0_BDH_LBKDIE_MASK
#define UARLTP_BDH_LBKDIE_SHIFT	UART0_BDH_LBKDIE_SHIFT
#define UARLTP_BDL_SBR_MASK	UART0_BDL_SBR_MASK
#define UARLTP_BDL_SBR_SHIFT	UART0_BDL_SBR_SHIFT
#define UARLTP_BDL_SBR(x)	UART0_BDL_SBR(x)
#define UARLTP_C1_PT_MASK	UART0_C1_PT_MASK
#define UARLTP_C1_PT_SHIFT	UART0_C1_PT_SHIFT
#define UARLTP_C1_PE_MASK	UART0_C1_PE_MASK
#define UARLTP_C1_PE_SHIFT	UART0_C1_PE_SHIFT
#define UARLTP_C1_ILT_MASK	UART0_C1_ILT_MASK
#define UARLTP_C1_ILT_SHIFT	UART0_C1_ILT_SHIFT
#define UARLTP_C1_WAKE_MASK	UART0_C1_WAKE_MASK
#define UARLTP_C1_WAKE_SHIFT	UART0_C1_WAKE_SHIFT
#define UARLTP_C1_M_MASK	UART0_C1_M_MASK
#define UARLTP_C1_M_SHIFT	UART0_C1_M_SHIFT
#define UARLTP_C1_RSRC_MASK	UART0_C1_RSRC_MASK
#define UARLTP_C1_RSRC_SHIFT	UART0_C1_RSRC_SHIFT
#define UARLTP_C1_DOZEEN_MASK	UART0_C1_DOZEEN_MASK
#define UARLTP_C1_DOZEEN_SHIFT	UART0_C1_DOZEEN_SHIFT
#define UARLTP_C1_LOOPS_MASK	UART0_C1_LOOPS_MASK
#define UARLTP_C1_LOOPS_SHIFT	UART0_C1_LOOPS_SHIFT
#define UARLTP_C2_SBK_MASK	UART0_C2_SBK_MASK
#define UARLTP_C2_SBK_SHIFT	UART0_C2_SBK_SHIFT
#define UARLTP_C2_RWU_MASK	UART0_C2_RWU_MASK
#define UARLTP_C2_RWU_SHIFT	UART0_C2_RWU_SHIFT
#define UARLTP_C2_RE_MASK	UART0_C2_RE_MASK

#define UARTLP_C2_RE_SHIFT	UART0_C2_RE_SHIFT
#define UARTLP_C2_TE_MASK	UART0_C2_TE_MASK
#define UARTLP_C2_TE_SHIFT	UART0_C2_TE_SHIFT
#define UARTLP_C2_ILIE_MASK	UART0_C2_ILIE_MASK
#define UARTLP_C2_ILIE_SHIFT	UART0_C2_ILIE_SHIFT
#define UARTLP_C2_RIE_MASK	UART0_C2_RIE_MASK
#define UARTLP_C2_RIE_SHIFT	UART0_C2_RIE_SHIFT
#define UARTLP_C2_TCIE_MASK	UART0_C2_TCIE_MASK
#define UARTLP_C2_TCIE_SHIFT	UART0_C2_TCIE_SHIFT
#define UARTLP_C2_TIE_MASK	UART0_C2_TIE_MASK
#define UARTLP_C2_TIE_SHIFT	UART0_C2_TIE_SHIFT
#define UARTLP_S1_PF_MASK	UART0_S1_PF_MASK
#define UARTLP_S1_PF_SHIFT	UART0_S1_PF_SHIFT
#define UARTLP_S1_FE_MASK	UART0_S1_FE_MASK
#define UARTLP_S1_FE_SHIFT	UART0_S1_FE_SHIFT
#define UARTLP_S1_NF_MASK	UART0_S1_NF_MASK
#define UARTLP_S1_NF_SHIFT	UART0_S1_NF_SHIFT
#define UARTLP_S1_OR_MASK	UART0_S1_OR_MASK
#define UARTLP_S1_OR_SHIFT	UART0_S1_OR_SHIFT
#define UARTLP_S1_IDLE_MASK	UART0_S1_IDLE_MASK
#define UARTLP_S1_IDLE_SHIFT	UART0_S1_IDLE_SHIFT
#define UARTLP_S1_RDRF_MASK	UART0_S1_RDRF_MASK
#define UARTLP_S1_RDRF_SHIFT	UART0_S1_RDRF_SHIFT
#define UARTLP_S1_TC_MASK	UART0_S1_TC_MASK
#define UARTLP_S1_TC_SHIFT	UART0_S1_TC_SHIFT
#define UARTLP_S1_TDRE_MASK	UART0_S1_TDRE_MASK
#define UARTLP_S1_TDRE_SHIFT	UART0_S1_TDRE_SHIFT
#define UARTLP_S2_RAF_MASK	UART0_S2_RAF_MASK
#define UARTLP_S2_RAF_SHIFT	UART0_S2_RAF_SHIFT

#define UARLTP_S2_LBKDE_MASK	UART0_S2_LBKDE_MASK
#define UARLTP_S2_LBKDE_SHIFT	UART0_S2_LBKDE_SHIFT
#define UARLTP_S2_BRK13_MASK	UART0_S2_BRK13_MASK
#define UARLTP_S2_BRK13_SHIFT	UART0_S2_BRK13_SHIFT
#define UARLTP_S2_RWUID_MASK	UART0_S2_RWUID_MASK
#define UARLTP_S2_RWUID_SHIFT	UART0_S2_RWUID_SHIFT
#define UARLTP_S2_RXINV_MASK	UART0_S2_RXINV_MASK
#define UARLTP_S2_RXINV_SHIFT	UART0_S2_RXINV_SHIFT
#define UARLTP_S2_MSBF_MASK	UART0_S2_MSBF_MASK
#define UARLTP_S2_MSBF_SHIFT	UART0_S2_MSBF_SHIFT
#define UARLTP_S2_RXEDGIF_MASK	UART0_S2_RXEDGIF_MASK
#define UARLTP_S2_RXEDGIF_SHIFT	UART0_S2_RXEDGIF_SHIFT
#define UARLTP_S2_LBKDIF_MASK	UART0_S2_LBKDIF_MASK
#define UARLTP_S2_LBKDIF_SHIFT	UART0_S2_LBKDIF_SHIFT
#define UARLTP_C3_PEIE_MASK	UART0_C3_PEIE_MASK
#define UARLTP_C3_PEIE_SHIFT	UART0_C3_PEIE_SHIFT
#define UARLTP_C3_FEIE_MASK	UART0_C3_FEIE_MASK
#define UARLTP_C3_FEIE_SHIFT	UART0_C3_FEIE_SHIFT
#define UARLTP_C3_NEIE_MASK	UART0_C3_NEIE_MASK
#define UARLTP_C3_NEIE_SHIFT	UART0_C3_NEIE_SHIFT
#define UARLTP_C3_ORIE_MASK	UART0_C3_ORIE_MASK
#define UARLTP_C3_ORIE_SHIFT	UART0_C3_ORIE_SHIFT
#define UARLTP_C3_TXINV_MASK	UART0_C3_TXINV_MASK
#define UARLTP_C3_TXINV_SHIFT	UART0_C3_TXINV_SHIFT
#define UARLTP_C3_TXDIR_MASK	UART0_C3_TXDIR_MASK
#define UARLTP_C3_TXDIR_SHIFT	UART0_C3_TXDIR_SHIFT
#define UARLTP_C3_R9T8_MASK	UART0_C3_R9T8_MASK
#define UARLTP_C3_R9T8_SHIFT	UART0_C3_R9T8_SHIFT
#define UARLTP_C3_R8T9_MASK	UART0_C3_R8T9_MASK

#define UARTLP_C3_R8T9_SHIFT	UART0_C3_R8T9_SHIFT
#define UARTLP_DROTO_MASK	UART0_DROTO_MASK
#define UARTLP_DROTO_SHIFT	UART0_DROTO_SHIFT
#define UARTLP_DR1T1_MASK	UART0_DR1T1_MASK
#define UARTLP_DR1T1_SHIFT	UART0_DR1T1_SHIFT
#define UARTLP_DR2T2_MASK	UART0_DR2T2_MASK
#define UARTLP_DR2T2_SHIFT	UART0_DR2T2_SHIFT
#define UARTLP_DR3T3_MASK	UART0_DR3T3_MASK
#define UARTLP_DR3T3_SHIFT	UART0_DR3T3_SHIFT
#define UARTLP_DR4T4_MASK	UART0_DR4T4_MASK
#define UARTLP_DR4T4_SHIFT	UART0_DR4T4_SHIFT
#define UARTLP_DR5T5_MASK	UART0_DR5T5_MASK
#define UARTLP_DR5T5_SHIFT	UART0_DR5T5_SHIFT
#define UARTLP_DR6T6_MASK	UART0_DR6T6_MASK
#define UARTLP_DR6T6_SHIFT	UART0_DR6T6_SHIFT
#define UARTLP_DR7T7_MASK	UART0_DR7T7_MASK
#define UARTLP_DR7T7_SHIFT	UART0_DR7T7_SHIFT
#define UARTLP_MA1_MA_MASK	UART0_MA1_MA_MASK
#define UARTLP_MA1_MA_SHIFT	UART0_MA1_MA_SHIFT
#define UARTLP_MA1_MA(x)	UART0_MA1_MA(x)
#define UARTLP_MA2_MA_MASK	UART0_MA2_MA_MASK
#define UARTLP_MA2_MA_SHIFT	UART0_MA2_MA_SHIFT
#define UARTLP_MA2_MA(x)	UART0_MA2_MA(x)
#define UARTLP_C4_OSR_MASK	UART0_C4_OSR_MASK
#define UARTLP_C4_OSR_SHIFT	UART0_C4_OSR_SHIFT
#define UARTLP_C4_OSR(x)	UART0_C4_OSR(x)
#define UARTLP_C4_M10_MASK	UART0_C4_M10_MASK
#define UARTLP_C4_M10_SHIFT	UART0_C4_M10_SHIFT
#define UARTLP_C4_MAEN2_MASK	UART0_C4_MAEN2_MASK

```
#define UARTLP_C4_MAEN2_SHIFT           UART0_C4_MAEN2_SHIFT
#define UARTLP_C4_MAEN1_MASK              UART0_C4_MAEN1_MASK
#define UARTLP_C4_MAEN1_SHIFT             UART0_C4_MAEN1_SHIFT
#define UARTLP_C5_RESETDISC_MASK          UART0_C5_RESETDISC_MASK
#define UARTLP_C5_RESETDISC_SHIFT         UART0_C5_RESETDISC_SHIFT
#define UARTLP_C5_BOTHEDGE_MASK           UART0_C5_BOTHEDGE_MASK
#define UARTLP_C5_BOTHEDGE_SHIFT          UART0_C5_BOTHEDGE_SHIFT
#define UARTLP_C5_RDMAE_MASK              UART0_C5_RDMAE_MASK
#define UARTLP_C5_RDMAE_SHIFT             UART0_C5_RDMAE_SHIFT
#define UARTLP_C5_TDMAE_MASK              UART0_C5_TDMAE_MASK
#define UARTLP_C5_TDMAE_SHIFT             UART0_C5_TDMAE_SHIFT

#define UARTLP_BASES                     UARTLP_BASES
#define NV_FOPT_EZPORT_DIS_MASK          This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT         This_symbol_has_been_deprecated
#define ADC_BASES                        ADC_BASE_PTRS
#define CMP_BASES                        CMP_BASE_PTRS
#define DAC_BASES                        DAC_BASE_PTRS
#define DMA_BASES                        DMA_BASE_PTRS
#define DMAMUX_BASES                    DMAMUX_BASE_PTRS
#define FPTA_BASE_PTR                   FGPIOA_BASE_PTR
#define FPTA_BASE                        FGPIOA_BASE
#define FPTA                           FGPIOA
#define FPTB_BASE_PTR                   FGPIOB_BASE_PTR
#define FPTB_BASE                        FGPIOB_BASE
#define FPTB                           FGPIOB
#define FPTC_BASE_PTR                   FGPIOC_BASE_PTR
#define FPTC_BASE                        FGPIOC_BASE
#define FPTC                           FGPIOC
#define FPTD_BASE_PTR                   FGPIOD_BASE_PTR
```

#define FPTD_BASE	FGPIOD_BASE
#define FPTD	FGPIOD
#define FPTE_BASE_PTR	FGPIOE_BASE_PTR
#define FPTE_BASE	FGPIOE_BASE
#define FPTE	FGPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define FTFA_BASES	FTFA_BASE_PTRS
#define PTA_BASE_PTR	GPIOA_BASE_PTR
#define PTA_BASE	GPIOA_BASE
#define PTA	GPIOA
#define PTB_BASE_PTR	GPIOB_BASE_PTR
#define PTB_BASE	GPIOB_BASE
#define PTB	GPIOB
#define PTC_BASE_PTR	GPIOC_BASE_PTR
#define PTC_BASE	GPIOC_BASE
#define PTC	GPIOC
#define PTD_BASE_PTR	GPIOD_BASE_PTR
#define PTD_BASE	GPIOD_BASE
#define PTD	GPIOD
#define PTE_BASE_PTR	GPIOE_BASE_PTR
#define PTE_BASE	GPIOE_BASE
#define PTE	GPIOE
#define GPIO_BASES	GPIO_BASE_PTRS
#define I2C_BASES	I2C_BASE_PTRS
#define LLWU_BASES	LLWU_BASE_PTRS
#define LPTMR_BASES	LPTMR_BASE_PTRS
#define MCG_BASES	MCG_BASE_PTRS
#define MCM_BASES	MCM_BASE_PTRS
#define MTB_BASES	MTB_BASE_PTRS

```
#define MTBDWT_BASES          MTBDWT_BASE_PTRS
#define NV_BASES                NV_BASES
#define OSC_BASES               OSC_BASE_PTRS
#define PIT_BASES               PIT_BASE_PTRS
#define PMC_BASES               PMC_BASE_PTRS
#define PORT_BASES              PORT_BASE_PTRS
#define RCM_BASES               RCM_BASE_PTRS
#define ROM_BASES               ROM_BASE_PTRS
#define RTC_BASES               RTC_BASE_PTRS
#define SIM_BASES               SIM_BASE_PTRS
#define SMC_BASES               SMC_BASE_PTRS
#define SPI_BASES               SPI_BASE_PTRS
#define TPM_BASES               TPM_BASE_PTRS
#define TSI_BASES               TSI_BASE_PTRS
#define UART_BASES              UART_BASE_PTRS
#define UART0_BASES             UART0_BASE_PTRS
#define USB_BASES               USB_BASE_PTRS
#define LPTimer_IRQn            LPTMR0_IRQn
#define LPTimer_IRQHandler       LPTMR0_IRQHandler
#define LLW_IRQn                LLWU_IRQn
#define LLW_IRQHandler          LLWU_IRQHandler
```

```
/*
 * @}
 */
/* /* end of group Backward_Compatibility_Symbols */
```

```
#else /* #if !defined(MKL25Z4_H_) */
```

```

/* There is already included the same memory map. Check if it is compatible (has the same major
version) */

#if (MCU_MEM_MAP_VERSION != 0x0200u)

#ifndef MCU_MEM_MAP_SUPPRESS_VERSION_WARNING)
    #warning There are included two not compatible versions of memory maps. Please check possible
differences.

#endif /* (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING)) */

#endif /* (MCU_MEM_MAP_VERSION != 0x0200u) */

#ifndef MKL25Z4_H_

/* MKL25Z4.h, eof. */

/*
 * memory.h
 *
 * Created on: Oct 28, 2017
 * Author: harsi
 */

#endif /* INCLUDES_MEMORY_H_ */

#define INCLUDES_MEMORY_H_

/***********************/

/** Authors: Harsimransingh Bindra, Smitesh Modak
 * Date Edited: 10/01/2017
 *
 * File: memory.h
 *
 * Description: Header file for data manipulation
 * -cal_length
 * -my_memmove

```

```
* * -my_memcpy
* * -my_memset
* * -my_memzero
* * -my_reverse
* * -reserve_words
* * -free_words
*
*
* *
*****
header for function called "memory"*/
```

```
#include <stdio.h>
#include <stdint.h>

typedef enum status_memory{
    Pass,
    Invalid_pointer,
    Fail
}memory_status_t;

uint8_t cal_length(uint8_t *ptr);
/***
 * Function to calculate length of the string
 *
 *This function calculates the length of the string which is stored at the memory location pointed by the
pointer passed
 *
 * returns uint8_t(length)*/
```

```
uint8_t my_memmove(uint8_t *src, uint8_t *dst, size_t length);

/**

 * Function to move a data from one location to another
 *
 * This function is used to move data from one memory location to another.
 * overlapping of the two memory locations is checked and handled accordingly
 *
 * returns uint8_t*/

uint8_t my_memcpy(uint8_t *src, uint8_t *dst, size_t length);

/**

 * Function to copy data from one location to another
 *
 * This function is used to copy data from one memory location to another
 * This function does not handle overlapping and data may be overwritten
 *
 * returns uint8_t*/

uint8_t my_memset(uint8_t *src, size_t length, uint8_t value);

/**

 * Function to set particular value to all given locations
 *
 * This function sets a particular value given as parameter to all given memory locations.
 *
 * returns uint8_t*/

uint8_t my_memzero(uint8_t *src, size_t length);

/**

 * function to set all values to 0 at given memory locations
 *
 * This function is a more specific version of memset function, it sets the value to 0 for all the memory
location given
```

```

*
* returns uint8_t*/
uint8_t my_reverse(uint8_t *src, size_t length);

/***
* Function to reverse data at given memory location
*
* This function reverses a particular array or a given string using pointer arithmetic
*
* returns uint8_t*/
uint32_t *reserve_words(size_t length);

/***
* function to reserve a particular memory location through dynamic memory allocation
*
* This function dynamically allocates memory to the pointer passed as the parameter
*
* returns uint32_t* */
void free_words(uint32_t *src);

/***
* Function to free dynamically allocated memory
*
* This function frees the memory which was dynamically allocated for the function. It avoids the
memory exhaustion
*
* returns void */
#endif /* INCLUDES_MEMORY_H_ */

#ifndef INCLUDES_CIRCBUF_H_

```

```
#define INCLUDES_CIRCBUF_H_
#include <stdint.h>

typedef enum CB_Enum_t{
    No_error=1,
    Full,
    Empty,
    Available,
    Null_pointer,
    No_length,
    Buffer_allocation_failure,
    Wrong_memory_access,
    Error
}CB_status;

typedef struct circular
{
    volatile uint8_t *buffer;
    volatile uint8_t *tail;
    volatile uint8_t *head;
    volatile uint8_t val;
    volatile uint8_t count;
    volatile uint8_t length;
    volatile uint8_t peek;
    CB_status status;
}CB_t;

CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val);
```

```

void memcopy(uint8_t *tail, uint8_t val);

CB_status CB_init(CB_t **cbuf,uint8_t length);

CB_status CB_buffer_remove_item(CB_t *cbuf/*, uint8_t *value*/);

CB_status CB_is_full(CB_t *cbuf);

CB_status CB_is_empty(CB_t *cbuf);

CB_status CB_peek(CB_t *cbuf, uint8_t position);

CB_status CB_destroy(CB_t *cbuf);

#endif /* INCLUDES_CIRCBUF_H_ */

//*******************************************************************************/

* @file core_cmlInstr.h
* @brief CMSIS Cortex-M Core Instruction Access Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*
*****/



/* Copyright (c) 2009 - 2014 ARM LIMITED

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#ifndef __CORE_CMINSTR_H  
#define __CORE_CMINSTR_H
```

```
/* ##### Core Instruction Access ##### */
```

```
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction Interface
Access to dedicated instructions

@{

*/
#ifndef __CC_ARM /*-----RealView Compiler -----*/
/* ARM armcc specific functions */

#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif

/** \brief No Operation

No Operation does nothing. This instruction can be used for code alignment purposes.

*/
#define __NOP          __nop

/** \brief Wait For Interrupt

Wait For Interrupt is a hint instruction that suspends execution
until one of a number of events occurs.

*/
#define __WFI          __wfi

/** \brief Wait For Event
```

Wait For Event is a hint instruction that permits the processor to enter a low-power state until one of a number of events occurs.

*/

```
#define __WFE           __wfe
```

```
/** \brief Send Event
```

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

*/

```
#define __SEV           __sev
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the instruction has been completed.

*/

```
#define __ISB() do {\\
    __schedule_barrier();\\
    __isb(0xF);\\
    __schedule_barrier();\\
} while (0)
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

```
*/  
  
#define __DSB() do {\  
    __schedule_barrier();\  
    __dsb(0xF);\  
    __schedule_barrier();\  
} while (0)
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before and after the instruction, without ensuring their completion.

```
*/  
  
#define __DMB() do {\  
    __schedule_barrier();\  
    __dmb(0xF);\  
    __schedule_barrier();\  
} while (0)
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

```
\param [in]  value Value to reverse  
\return      Reversed value  
*/  
  
#define __REV          __rev
```

```
/** \brief Reverse byte order (16 bit)
```

This function reverses the byte order in two unsigned short values.

```
\param [in] value Value to reverse  
\return Reversed value  
*/  
  
#ifndef __NO_EMBEDDED_ASM  
  
__attribute__((section(".rev16_text"))) __STATIC_INLINE __ASM uint32_t __REV16(uint32_t value)  
{  
    rev16 r0, r0  
    bx lr  
}  
  
#endif
```

```
/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in] value Value to reverse  
\return Reversed value  
*/  
  
#ifndef __NO_EMBEDDED_ASM  
  
__attribute__((section(".revsh_text"))) __STATIC_INLINE __ASM int32_t __REVSH(int32_t value)  
{  
    revsh r0, r0  
    bx lr  
}  
  
#endif
```

```
/** \brief Rotate Right in unsigned value (32 bit)
```

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```
\param [in]  value Value to rotate  
\param [in]  value Number of Bits to rotate  
\return      Rotated value  
*/
```

```
#define __ROR          __ror
```

```
/** \brief Breakpoint
```

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

```
\param [in]  value is ignored by the processor.
```

If required, a debugger can use it to store additional information about the breakpoint.

```
*/  
#define __BKPT(value)      __breakpoint(value)
```

```
/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

```

\param [in]  value Value to reverse

\return      Reversed value

*/
#ifndef __RBIT_H
#define __RBIT_H

#if   (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
#define __RBIT          __rbit
#else
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t value)
{
    uint32_t result;
    int32_t s = 4 /*sizeof(v)*/ * 8 - 1; // extra shift needed at end

    result = value;           // r will be reversed bits of v; first get LSB of v
    for (value >>= 1; value; value >>= 1)
    {
        result <<= 1;
        result |= value & 1;
        s--;
    }
    result <<= s;           // shift when v's highest bits are zero
    return(result);
}

#endif

/** \brief Count leading zeros

```

This function counts the number of leading zeros of a data value.

```
\param [in] value Value to count the leading zeros  
\return number of leading zeros in value  
*/  
#define __CLZ                    __clz
```

```
#if    (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
```

```
/** \brief LDR Exclusive (8 bit)
```

This function executes a exclusive LDR instruction for 8 bit value.

```
\param [in] ptr Pointer to data  
\return value of type uint8_t at (*ptr)  
*/  
#define __LDREXB(ptr)            ((uint8_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (16 bit)
```

This function executes a exclusive LDR instruction for 16 bit values.

```
\param [in] ptr Pointer to data  
\return value of type uint16_t at (*ptr)  
*/  
#define __LDREXBH(ptr)            ((uint16_t) __ldrex(ptr))
```

```
/** \brief LDR Exclusive (32 bit)
```

This function executes a exclusive LDR instruction for 32 bit values.

```
\param [in]  ptr  Pointer to data
\return     value of type uint32_t at (*ptr)
*/
#define __LDREXW(ptr)          ((uint32_t ) __ldrex(ptr))
```

```
/** \brief STR Exclusive (8 bit)
```

This function executes a exclusive STR instruction for 8 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return     0  Function succeeded
\return     1  Function failed
*/
#define __STREXB(value, ptr)      __strex(value, ptr)
```

```
/** \brief STR Exclusive (16 bit)
```

This function executes a exclusive STR instruction for 16 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return     0  Function succeeded
\return     1  Function failed
```

```
*/  
#define __STREXH(value, ptr)      __strex(value, ptr)
```

```
/** \brief STR Exclusive (32 bit)
```

This function executes a exclusive STR instruction for 32 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
\return 0 Function succeeded  
\return 1 Function failed
```

```
*/  
#define __STREXW(value, ptr)      __strex(value, ptr)
```

```
/** \brief Remove the exclusive lock
```

This function removes the exclusive lock which is created by LDREX.

```
*/  
#define __CLREX      __clrex
```

```
/** \brief Signed Saturate
```

This function saturates a signed value.

```
\param [in] value Value to be saturated
```

```
\param [in] sat Bit position to saturate to (1..32)
\return Saturated value
*/
#define __SSAT __ssat
```

```
/** \brief Unsigned Saturate
```

This function saturates an unsigned value.

```
\param [in] value Value to be saturated
\param [in] sat Bit position to saturate to (0..31)
\return Saturated value
*/
#define __USAT __usat
```

```
/** \brief Rotate Right with Extend (32 bit)
```

This function moves each bit of a bitstring right by one bit.

The carry input is shifted in at the left end of the bitstring.

```
\param [in] value Value to rotate
\return Rotated value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".rrx_text"))) __STATIC_INLINE __ASM uint32_t __RRX(uint32_t value)
{
    rrx r0, r0
```

```
bx lr
```

```
}
```

```
#endif
```

```
/** \brief LDRT Unprivileged (8 bit)
```

This function executes a Unprivileged LDRT instruction for 8 bit value.

```
\param [in]  ptr  Pointer to data
```

```
\return     value of type uint8_t at (*ptr)
```

```
*/
```

```
#define __LDRBT(ptr)          ((uint8_t) __ldrt(ptr))
```

```
/** \brief LDRT Unprivileged (16 bit)
```

This function executes a Unprivileged LDRT instruction for 16 bit values.

```
\param [in]  ptr  Pointer to data
```

```
\return     value of type uint16_t at (*ptr)
```

```
*/
```

```
#define __LDRHT(ptr)          ((uint16_t) __ldrt(ptr))
```

```
/** \brief LDRT Unprivileged (32 bit)
```

This function executes a Unprivileged LDRT instruction for 32 bit values.

```
\param [in]  ptr  Pointer to data
\return     value of type uint32_t at (*ptr)
*/
#define __LDRT(ptr)          ((uint32_t ) __ldrt(ptr))
```

```
/** \brief STRT Unprivileged (8 bit)
```

This function executes a Unprivileged STRT instruction for 8 bit values.

```
\param [in] value  Value to store
\param [in]  ptr  Pointer to location
*/
#define __STRBT(value, ptr)      __strt(value, ptr)
```

```
/** \brief STRT Unprivileged (16 bit)
```

This function executes a Unprivileged STRT instruction for 16 bit values.

```
\param [in] value  Value to store
\param [in]  ptr  Pointer to location
*/
#define __STRHT(value, ptr)      __strt(value, ptr)
```

```
/** \brief STRT Unprivileged (32 bit)
```

This function executes a Unprivileged STRT instruction for 32 bit values.

```

    \param [in] value Value to store

    \param [in] ptr Pointer to location

*/
#define __STRT(value, ptr)      __strt(value, ptr)

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#ifndef __GNUC__
/*----- GNU Compiler -----*/
/* GNU gcc specific functions */

/* Define macros for porting to both thumb1 and thumb2.

 * For thumb1, use low register (r0-r7), specified by constraint "l"
 * Otherwise, use general registers, specified by constraint "r" */

#if defined(__thumb__) && !defined(__thumb2__)
#define __CMSIS_GCC_OUT_REG(r) "=l" (r)
#define __CMSIS_GCC_USE_REG(r) "l" (r)
#else
#define __CMSIS_GCC_OUT_REG(r) "=r" (r)
#define __CMSIS_GCC_USE_REG(r) "r" (r)
#endif

/** \brief No Operation

```

No Operation does nothing. This instruction can be used for code alignment purposes.

```

*/
__attribute__((always_inline)) __STATIC_INLINE void __NOP(void)
{
```

```
__ASM volatile ("nop");  
}
```

```
/** \brief Wait For Interrupt
```

Wait For Interrupt is a hint instruction that suspends execution until one of a number of events occurs.

```
*/  
  
__attribute__((always_inline)) __STATIC_INLINE void __WFI(void)  
{  
    __ASM volatile ("wfi");  
}
```

```
/** \brief Wait For Event
```

Wait For Event is a hint instruction that permits the processor to enter a low-power state until one of a number of events occurs.

```
*/  
  
__attribute__((always_inline)) __STATIC_INLINE void __WFE(void)  
{  
    __ASM volatile ("wfe");  
}
```

```
/** \brief Send Event
```

Send Event is a hint instruction. It causes an event to be signaled to the CPU.

```
*/  
__attribute__((always_inline)) __STATIC_INLINE void __SEV(void)  
{  
    __ASM volatile ("sev");  
}
```

```
/** \brief Instruction Synchronization Barrier
```

Instruction Synchronization Barrier flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the instruction has been completed.

```
*/  
__attribute__((always_inline)) __STATIC_INLINE void __ISB(void)  
{  
    __ASM volatile ("isb 0xF":::"memory");  
}
```

```
/** \brief Data Synchronization Barrier
```

This function acts as a special kind of Data Memory Barrier.

It completes when all explicit memory accesses before this instruction complete.

```
*/  
__attribute__((always_inline)) __STATIC_INLINE void __DSB(void)  
{  
    __ASM volatile ("dsb 0xF":::"memory");  
}
```

```
/** \brief Data Memory Barrier
```

This function ensures the apparent order of the explicit memory operations before and after the instruction, without ensuring their completion.

```
*/  
  
__attribute__((always_inline)) __STATIC_INLINE void __DMB(void)  
{  
    __ASM volatile ("dmb 0xF:::memory");  
}
```

```
/** \brief Reverse byte order (32 bit)
```

This function reverses the byte order in integer value.

```
\param [in]  value  Value to reverse  
\return      Reversed value  
*/  
  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV(uint32_t value)  
{  
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 5)  
    return __builtin_bswap32(value);  
#else  
    uint32_t result;  
  
    __ASM volatile ("rev %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));  
    return(result);  
#endif
```

```
}
```

```
/** \brief Reverse byte order (16 bit)
```

This function reverses the byte order in two unsigned short values.

```
\param [in] value Value to reverse
\return Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV16(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("rev16 %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
}

/** \brief Reverse byte order in signed short value
```

This function reverses the byte order in a signed short value with sign extension to integer.

```
\param [in] value Value to reverse
\return Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE int32_t __REVSH(int32_t value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
```

```

return (short) __builtin_bswap16(value);

#else

uint32_t result;

__ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value) );
return(result);

#endif

}

```

/** \brief Rotate Right in unsigned value (32 bit)

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```

\param [in]  value Value to rotate
\param [in]  value Number of Bits to rotate
\return      Rotated value
*/

```

```

__attribute__((always_inline)) __STATIC_INLINE uint32_t __ROR(uint32_t op1, uint32_t op2)
{
    return (op1 >> op2) | (op1 << (32 - op2));
}

```

/** \brief Breakpoint

This function causes the processor to enter Debug state.

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

\param [in] value is ignored by the processor.

If required, a debugger can use it to store additional information about the breakpoint.

```
*/  
#define __BKPT(value)           __ASM volatile ("bkpt "#value)
```

```
/** \brief Reverse bit order of value
```

This function reverses the bit order of the given value.

```
\param [in] value Value to reverse  
\return      Reversed value  
*/  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t value)  
{  
    uint32_t result;  
  
    #if   (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)  
        __ASM volatile ("rbit %0, %1" : "=r" (result) : "r" (value));  
    #else  
        int32_t s = 4 /*sizeof(v)*/ * 8 - 1; // extra shift needed at end  
  
        result = value;          // r will be reversed bits of v; first get LSB of v  
        for (value >>= 1; value; value >>= 1)  
        {  
            result <= 1;
```

```

result |= value & 1;

s--;
}

result <<= s;           // shift when v's highest bits are zero

#endif

return(result);

}

```

/** \brief Count leading zeros

This function counts the number of leading zeros of a data value.

```

\param [in] value Value to count the leading zeros

\return number of leading zeros in value

*/
#define __CLZ      __builtin_clz

```

#if (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)

/** \brief LDR Exclusive (8 bit)

This function executes a exclusive LDR instruction for 8 bit value.

```

\param [in] ptr Pointer to data

\return value of type uint8_t at (*ptr)

*/
__attribute__((always_inline)) __STATIC_INLINE uint8_t __LDREXB(volatile uint8_t *addr)

```

```

{

    uint32_t result;

#ifndef __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexb %0, %1" : "=r" (result) : "Q" (*addr));
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.

*/
    __ASM volatile ("ldrexb %0, [%1]" : "=r" (result) : "r" (addr) : "memory");
#endif

    return ((uint8_t) result); /* Add explicit type cast here */
}

```

/** \brief LDR Exclusive (16 bit)

This function executes a exclusive LDR instruction for 16 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint16_t at (*ptr)

*/
__attribute__((always_inline)) __STATIC_INLINE uint16_t __LDREXH(volatile uint16_t *addr)
{
    uint32_t result;

#ifndef __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexb %0, %1" : "=r" (result) : "Q" (*addr));
#else

```

```

/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.

*/
__ASM volatile ("ldrexh %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );
#endif

return ((uint16_t) result); /* Add explicit type cast here */
}

```

/** \brief LDR Exclusive (32 bit)

This function executes a exclusive LDR instruction for 32 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint32_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __LDREXW(volatile uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrex %0, %1" : "=r" (result) : "Q" (*addr) );
    return(result);
}

```

/** \brief STR Exclusive (8 bit)

This function executes a exclusive STR instruction for 8 bit values.

```

    \param [in] value Value to store
    \param [in] ptr Pointer to location
    \return 0 Function succeeded
    \return 1 Function failed
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXB(uint8_t value, volatile uint8_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value));
    return(result);
}

```

/** \brief STR Exclusive (16 bit)

This function executes a exclusive STR instruction for 16 bit values.

```

    \param [in] value Value to store
    \param [in] ptr Pointer to location
    \return 0 Function succeeded
    \return 1 Function failed
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXH(uint16_t value, volatile uint16_t
*addr)
{
    uint32_t result;

```

```
    __ASM volatile ("strexh %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" ((uint32_t)value) );
    return(result);
}
```

```
/** \brief STR Exclusive (32 bit)
```

This function executes a exclusive STR instruction for 32 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
\return 0 Function succeeded
\return 1 Function failed
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXW(uint32_t value, volatile uint32_t
*addr)
{
    uint32_t result;

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) : "r" (value) );
    return(result);
}
```

```
/** \brief Remove the exclusive lock
```

This function removes the exclusive lock which is created by LDREX.

```
*/
```

```
__attribute__((always_inline)) __STATIC_INLINE void __CLREX(void)
{
    __ASM volatile ("clrex" ::: "memory");
}
```

```
/** \brief Signed Saturate
```

This function saturates a signed value.

```
\param [in] value Value to be saturated
\param [in] sat Bit position to saturate to (1..32)
\return Saturated value
*/
#define __SSAT(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})
```

```
/** \brief Unsigned Saturate
```

This function saturates an unsigned value.

```
\param [in] value Value to be saturated
\param [in] sat Bit position to saturate to (0..31)
\return Saturated value
```

```

*/
#define __USAT(ARG1,ARG2) \
({           \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

```

/** \brief Rotate Right with Extend (32 bit)

This function moves each bit of a bitstring right by one bit.

The carry input is shifted in at the left end of the bitstring.

```

\param [in]  value Value to rotate
\return      Rotated value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RRX(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("rrx %0, %1" : __CMSIS_GCC_OUT_REG (result) : __CMSIS_GCC_USE_REG (value));
    return(result);
}

```

/** \brief LDRT Unprivileged (8 bit)

This function executes a Unprivileged LDRT instruction for 8 bit value.

```

\param [in]  ptr  Pointer to data
\return     value of type uint8_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint8_t __LDRBT(volatile uint8_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrbt %0, %1" : "=r" (result) : "Q" (*addr));
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
   accepted by assembler. So has to use following less efficient pattern.
*/
    __ASM volatile ("ldrbt %0, [%1]" : "=r" (result) : "r" (addr) : "memory");
#endif
    return ((uint8_t) result); /* Add explicit type cast here */
}

```

/** \brief LDRT Unprivileged (16 bit)

This function executes a Unprivileged LDRT instruction for 16 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint16_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint16_t __LDRHT(volatile uint16_t *addr)
{

```

```

uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrht %0, %1" : "=r" (result) : "Q" (*addr));
#else
/* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
accepted by assembler. So has to use following less efficient pattern.

*/
    __ASM volatile ("ldrht %0, [%1]" : "=r" (result) : "r" (addr) : "memory");
#endif
return ((uint16_t) result); /* Add explicit type cast here */
}

```

/** \brief LDRT Unprivileged (32 bit)

This function executes a Unprivileged LDRT instruction for 32 bit values.

```

\param [in]  ptr  Pointer to data
\return     value of type uint32_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __LDRT(volatile uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrt %0, %1" : "=r" (result) : "Q" (*addr));
    return(result);
}

```

```
/** \brief STRT Unprivileged (8 bit)
```

This function executes a Unprivileged STRT instruction for 8 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
*/  


```
__attribute__((always_inline)) __STATIC_INLINE void __STRBT(uint8_t value, volatile uint8_t *addr)
{
 __ASM volatile ("strbt %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value));
}
```


```

```
/** \brief STRT Unprivileged (16 bit)
```

This function executes a Unprivileged STRT instruction for 16 bit values.

```
\param [in] value Value to store  
\param [in] ptr Pointer to location  
*/  


```
__attribute__((always_inline)) __STATIC_INLINE void __STRHT(uint16_t value, volatile uint16_t *addr)
{
 __ASM volatile ("strht %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value));
}
```


```

```
/** \brief STRT Unprivileged (32 bit)
```

This function executes a Unprivileged STRT instruction for 32 bit values.

```
\param [in] value Value to store
\param [in] ptr Pointer to location
*/
__attribute__((always_inline)) __STATIC_INLINE void __STRT(uint32_t value, volatile uint32_t *addr)
{
    __ASM volatile ("strt %1, %0" : "=Q" (*addr) : "r" (value));
}

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#if defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#if defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING ccm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
*/
```

```

*/



#ifndef __CORE_CM0PLUS_H
#define __CORE_CM0PLUS_H

/* Cosmic specific functions */

#include <cmsis_csm.h>

#endif /* __CORE_CM0PLUS_H */

/* @file core_cm0plus.h
 * @brief CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
 * @version V4.10
 * @date 18. March 2015
 *
 * @note
 *
 */
/* Copyright (c) 2009 - 2015 ARM LIMITED
 */

```

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 ARM 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 COPYRIGHT HOLDERS AND 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.

-----*/

```
#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA check */
#endif
```

```
#ifndef __CORE_CM0PLUS_H_GENERIC
```

```
#define __CORE_CM0PLUS_H_GENERIC
```

```
#ifdef __cplusplus
```

```
extern "C" {
```

```
#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions
```

```
CMSIS violates the following MISRA-C:2004 rules:
```

```
\li Required Rule 8.5, object/function definition in header file.<br>
```

```
Function definitions in header files are used to allow 'inlining'.
```

```
\li Required Rule 18.4, declaration of union type or object of union type: '{...}'.<br>
```

```
Unions are used for effective representation of core registers.
```

```
\li Advisory Rule 19.7, Function-like macro defined.<br>
```

```
Function-like macros are used to allow more efficient code.
```

```
*/
```

```
*****
```

```
* CMSIS definitions
```

```
***** /
```

```
/** \ingroup Cortex-M0+
```

```
@{
```

```
*/
```

```
/* CMSIS CM0P definitions */
```

```
#define __CM0PLUS_CMSIS_VERSION_MAIN (0x04) /*!< [31:16] CMSIS HAL main  
version */
```

```
#define __CM0PLUS_CMSIS_VERSION_SUB (0x00) /*!< [15:0] CMSIS HAL sub version  
*/
```

```
#define __CM0PLUS_CMSIS_VERSION ((__CM0PLUS_CMSIS_VERSION_MAIN << 16) | \
```

```

    __CM0PLUS_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */

#define __CORTEX_M          (0x00)           /*!< Cortex-M Core */

#if defined (__CC_ARM)
#define __ASM      __asm           /*!< asm keyword for ARM Compiler */
#define __INLINE   __inline        /*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)
#define __ASM      __asm           /*!< asm keyword for GNU Compiler */
#define __INLINE   inline          /*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)
#define __ASM      __asm           /*!< asm keyword for IAR Compiler */
#define __INLINE   inline          /*!< inline keyword for IAR Compiler. Only available in
High optimization mode! */
#define __STATIC_INLINE static inline

#elif defined (__TMS470__)
#define __ASM      __asm           /*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#elif defined (__TASKING__)
#define __ASM      __asm           /*!< asm keyword for TASKING Compiler */
#define __INLINE   inline          /*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

```

```

#define __packed
#define __ASM      _asm          /*!< asm keyword for COSMIC Compiler */
#define __INLINE    inline        /*use -pc99 on compile line !< inline keyword for
COSMIC Compiler */
#define __STATIC_INLINE static inline

#endif

/** __FPU_USED indicates whether an FPU is used or not.
This core does not support an FPU at all
*/
#define __FPU_USED    0

#if defined ( __CC_ARM )
#if defined __TARGET_FPU_VFP
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __GNUC__ )
#if defined (__VFP_FP__) && !defined(__SOFTFP__)
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __ICCARM__ )
#if defined __ARMVFP__
#warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

```

```
#elif defined ( __TMS470__ )

#if defined __TI__VFP_SUPPORT_____
    #warning "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __TASKING__ )

#if defined __FPU_VFP__
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __CSMC__ )          /* Cosmic */
#if ( __CSMC__ & 0x400)           // FPU present for parser
    #error "Compiler generates FPU instructions for a device without an FPU (check __FPU_PRESENT)"
#endif
#endif

#include <stdint.h>          /* standard types definitions      */
#include <core_cmInstr.h>        /* Core Instruction Access       */
#include <core_cmFunc.h>         /* Core Function Access          */

#endif /* __cplusplus

#endif /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC
```

```
#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

#ifndef __cplusplus
extern "C" {

#endif

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
#ifndef __CM0PLUS_REV
#define __CM0PLUS_REV      0x0000
#warning "__CM0PLUS_REV not defined in device header file; using default!"
#endif

#ifndef __MPU_PRESENT
#define __MPU_PRESENT      0
#warning "__MPU_PRESENT not defined in device header file; using default!"
#endif

#ifndef __VTOR_PRESENT
#define __VTOR_PRESENT      0
#warning "__VTOR_PRESENT not defined in device header file; using default!"
#endif

#ifndef __NVIC_PRIO_BITS
#define __NVIC_PRIO_BITS      2
#warning "__NVIC_PRIO_BITS not defined in device header file; using default!"
#endif
}
```

```

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0
#warning "__Vendor_SysTickConfig not defined in device header file; using default!"
#endif
#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
\defgroup CMSIS_glob_defs CMSIS Global Defines

<strong>IO Type Qualifiers</strong> are used
\li to specify the access to peripheral variables.
\li for automatic generation of peripheral register debug information.
*/
#endif

#ifndef __cplusplus
#define __I volatile /*!< Defines 'read only' permissions */
#ifndef __I
#define __I volatile const /*!< Defines 'read only' permissions */
#endif
#define __O volatile /*!< Defines 'write only' permissions */
#define __IO volatile /*!< Defines 'read / write' permissions */
#endif

/*}@} end of group Cortex-M0+ */

*****
*      Register Abstraction
Core Register contain:

```

- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core MPU Register

******/

```
/** \defgroup CMSIS_core_register Defines and Type Definitions
 \brief Type definitions and defines for Cortex-M processor based devices.
 */
```

```
/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CORE Status and Control Registers
 \brief Core Register type definitions.
 @{
 */

```

```
/** \brief Union type to access the Application Program Status Register (APSR).
```

```
*/

```

```
typedef union
{
    struct
    {
        uint32_t _reserved0:28;      /*!< bit: 0..27 Reserved          */
        uint32_t V:1;              /*!< bit: 28 Overflow condition code flag   */
        uint32_t C:1;              /*!< bit: 29 Carry condition code flag    */
        uint32_t Z:1;              /*!< bit: 30 Zero condition code flag     */
        uint32_t N:1;              /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access          */
    uint32_t w;                /*!< Type used for word access           */
}
```

```

} APSR_Type;

/* APSR Register Definitions */

#define APSR_N_Pos           31          /*!< APSR: N Position */
#define APSR_N_Msk          (1UL << APSR_N_Pos) /*!< APSR: N Mask */

#define APSR_Z_Pos           30          /*!< APSR: Z Position */
#define APSR_Z_Msk          (1UL << APSR_Z_Pos) /*!< APSR: Z Mask */

#define APSR_C_Pos           29          /*!< APSR: C Position */
#define APSR_C_Msk          (1UL << APSR_C_Pos) /*!< APSR: C Mask */

#define APSR_V_Pos           28          /*!< APSR: V Position */
#define APSR_V_Msk          (1UL << APSR_V_Pos) /*!< APSR: V Mask */

```

/** \brief Union type to access the Interrupt Program Status Register (IPSR).

```

*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} IPSR_Type;

```

/* IPSR Register Definitions */

```

#define IPSR_ISR_Pos          0                  /*!< IPSR: ISR Position */
#define IPSR_ISR_Msk          (0x1FFUL/*<< IPSR_ISR_Pos*/)      /*!< IPSR: ISR Mask */

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
*/
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0..8 Exception number */
        uint32_t _reserved0:15;   /*!< bit: 9..23 Reserved */
        uint32_t T:1;             /*!< bit: 24 Thumb bit (read 0) */
        uint32_t _reserved1:3;    /*!< bit: 25..27 Reserved */
        uint32_t V:1;             /*!< bit: 28 Overflow condition code flag */
        uint32_t C:1;             /*!< bit: 29 Carry condition code flag */
        uint32_t Z:1;             /*!< bit: 30 Zero condition code flag */
        uint32_t N:1;             /*!< bit: 31 Negative condition code flag */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} xPSR_Type;

/* xPSR Register Definitions */

#define xPSR_N_Pos            31                 /*!< xPSR: N Position */
#define xPSR_N_Msk            (1UL << xPSR_N_Pos)      /*!< xPSR: N Mask */

#define xPSR_Z_Pos            30                 /*!< xPSR: Z Position */
#define xPSR_Z_Msk            (1UL << xPSR_Z_Pos)      /*!< xPSR: Z Mask */

```

```

#define xPSR_C_Pos           29          /*!< xPSR: C Position */
#define xPSR_C_Msk          (1UL << xPSR_C_Pos)    /*!< xPSR: C Mask */

#define xPSR_V_Pos           28          /*!< xPSR: V Position */
#define xPSR_V_Msk          (1UL << xPSR_V_Pos)    /*!< xPSR: V Mask */

#define xPSR_T_Pos           24          /*!< xPSR: T Position */
#define xPSR_T_Msk          (1UL << xPSR_T_Pos)    /*!< xPSR: T Mask */

#define xPSR_ISR_Pos          0          /*!< xPSR: ISR Position */
#define xPSR_ISR_Msk        (0x1FFUL /*<< xPSR_ISR_Pos*/) /*!< xPSR: ISR Mask */

```

/** \brief Union type to access the Control Registers (CONTROL).

```

*/
typedef union
{
    struct
    {
        uint32_t nPRIV:1;           /*!< bit: 0 Execution privilege in Thread mode */
        uint32_t SPSEL:1;          /*!< bit: 1 Stack to be used */
        uint32_t _reserved1:30;     /*!< bit: 2..31 Reserved */
    } b;                      /*!< Structure used for bit access */
    uint32_t w;                /*!< Type used for word access */
} CONTROL_Type;

```

/* CONTROL Register Definitions */

```
#define CONTROL_SPSEL_Pos      1          /*!< CONTROL: SPSEL Position */
```

```

#define CONTROL_SPSEL_Msk          (1UL << CONTROL_SPSEL_Pos)      /*!< CONTROL: SPSEL
Mask */

#define CONTROL_nPRIV_Pos          0                                /*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk          (1UL /*<< CONTROL_nPRIV_Pos*/)    /*!< CONTROL:
nPRIV Mask */

/*@} end of group CMSIS_CORE */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
@{
*/

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[1];           /*!< Offset: 0x000 (R/W) Interrupt Set Enable Register */
    uint32_t RESERVED0[31];
    __IO uint32_t ICER[1];           /*!< Offset: 0x080 (R/W) Interrupt Clear Enable Register */
    uint32_t RSERVED1[31];
    __IO uint32_t ISPR[1];           /*!< Offset: 0x100 (R/W) Interrupt Set Pending Register */
    uint32_t RESERVED2[31];
    __IO uint32_t ICPR[1];           /*!< Offset: 0x180 (R/W) Interrupt Clear Pending Register */
    uint32_t RESERVED3[31];
    uint32_t RESERVED4[64];
}

```

```

    __IO uint32_t IP[8];           /*!< Offset: 0x300 (R/W) Interrupt Priority Register      */
} NVIC_Type;

/*}@} end of group CMSIS_NVIC */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers

@{

*/

```

```

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct
{
    __I uint32_t CPUID;          /*!< Offset: 0x000 (R/ ) CPUID Base Register            */
    __IO uint32_t ICSR;          /*!< Offset: 0x004 (R/W) Interrupt Control and State Register */
}

#ifndef __VTOR_PRESENT
    __IO uint32_t VTOR;          /*!< Offset: 0x008 (R/W) Vector Table Offset Register      */
#else
    uint32_t RESERVED0;
#endif

#ifndef __AIRCR_PRESENT
    __IO uint32_t AIRCR;         /*!< Offset: 0x00C (R/W) Application Interrupt and Reset Control Register */
    __IO uint32_t SCR;           /*!< Offset: 0x010 (R/W) System Control Register           */
    __IO uint32_t CCR;           /*!< Offset: 0x014 (R/W) Configuration Control Register     */
    uint32_t RESERVED1;

```

```

__IO uint32_t SHP[2];           /*!< Offset: 0x01C (R/W) System Handlers Priority Registers. [0] is
RESERVED */

__IO uint32_t SHCSR;           /*!< Offset: 0x024 (R/W) System Handler Control and State Register
*/

} SCB_Type;

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos    24           /*!< SCB CPUID:
IMPLEMENTER Position */

#define SCB_CPUID_IMPLEMENTER_Msk    (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos) /*!<
SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos       20           /*!< SCB CPUID: VARIANT Position
*/
#define SCB_CPUID_VARIANT_Msk       (0xFUL << SCB_CPUID_VARIANT_Pos) /*!< SCB CPUID:
VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos   16           /*!< SCB CPUID:
ARCHITECTURE Position */
#define SCB_CPUID_ARCHITECTURE_Msk   (0xFUL << SCB_CPUID_ARCHITECTURE_Pos) /*!<
SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos        4            /*!< SCB CPUID: PARTNO Position
*/
#define SCB_CPUID_PARTNO_Msk        (0xFFFFUL << SCB_CPUID_PARTNO_Pos) /*!< SCB
CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos      0            /*!< SCB CPUID: REVISION Position
*/
#define SCB_CPUID_REVISION_Msk      (0xFUL /*<< SCB_CPUID_REVISION_Pos*/) /*!<
SCB CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

```

#define SCB_ICSR_NMIPENDSET_Pos	31	/*!< SCB ICSR: NMIPENDSET Position */
#define SCB_ICSR_NMIPENDSET_Msk	(1UL << SCB_ICSR_NMIPENDSET_Pos)	/*!< SCB ICSR: NMIPENDSET Mask */
#define SCB_ICSR_PENDSVSET_Pos	28	/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk	(1UL << SCB_ICSR_PENDSVSET_Pos)	/*!< SCB ICSR: PENDSVSET Mask */
#define SCB_ICSR_PENDSVCLR_Pos	27	/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk	(1UL << SCB_ICSR_PENDSVCLR_Pos)	/*!< SCB ICSR: PENDSVCLR Mask */
#define SCB_ICSR_PENDSTSET_Pos	26	/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk	(1UL << SCB_ICSR_PENDSTSET_Pos)	/*!< SCB ICSR: PENDSTSET Mask */
#define SCB_ICSR_PENDSTCLR_Pos	25	/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk	(1UL << SCB_ICSR_PENDSTCLR_Pos)	/*!< SCB ICSR: PENDSTCLR Mask */
#define SCB_ICSR_ISRPREEMPT_Pos	23	/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk	(1UL << SCB_ICSR_ISRPREEMPT_Pos)	/*!< SCB ICSR: ISRPREEMPT Mask */
#define SCB_ICSR_ISRPENDING_Pos	22	/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk	(1UL << SCB_ICSR_ISRPENDING_Pos)	/*!< SCB ICSR: ISRPENDING Mask */

```

#define SCB_ICSR_VECTPENDING_Pos      12           /*!< SCB ICSR: VECTPENDING
Position */

#define SCB_ICSR_VECTPENDING_Msk     (0x1FFUL << SCB_ICSR_VECTPENDING_Pos)    /*!< SCB
ICSR: VECTPENDING Mask */

#define SCB_ICSR_VECTACTIVE_Pos      0            /*!< SCB ICSR: VECTACTIVE
Position */

#define SCB_ICSR_VECTACTIVE_Msk     (0x1FFUL /*<< SCB_ICSR_VECTACTIVE_Pos*/)   /*!< SCB
ICSR: VECTACTIVE Mask */

#if (__VTOR_PRESENT == 1)

/* SCB Interrupt Control State Register Definitions */

#define SCB_VTOR_TBLOFF_Pos        8            /*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk       (0xFFFFFUL << SCB_VTOR_TBLOFF_Pos)    /*!< SCB VTOR:
TBLOFF Mask */

#endif

/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos      16          /*!< SCB AIRCR: VECTKEY Position */

#define SCB_AIRCR_VECTKEY_Msk     (0xFFFFFUL << SCB_AIRCR_VECTKEY_Pos)    /*!< SCB
AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos    16          /*!< SCB AIRCR: VECTKEYSTAT
Position */

#define SCB_AIRCR_VECTKEYSTAT_Msk  (0xFFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)  /*!< SCB
AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos     15          /*!< SCB AIRCR: ENDIANESS
Position */

#define SCB_AIRCR_ENDIANESS_Msk   (1UL << SCB_AIRCR_ENDIANESS_Pos)    /*!< SCB
AIRCR: ENDIANESS Mask */

```

```

#define SCB_AIRCR_SYSRESETREQ_Pos      2                  /*!< SCB AIRCR: SYSRESETREQ
Position */

#define SCB_AIRCR_SYSRESETREQ_Msk     (1UL << SCB_AIRCR_SYSRESETREQ_Pos)    /*!< SCB
AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos   1                  /*!< SCB AIRCR: VECTCLRACTIVE
Position */

#define SCB_AIRCR_VECTCLRACTIVE_Msk  (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)    /*!< SCB
AIRCR: VECTCLRACTIVE Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos        4                  /*!< SCB SCR: SEVONPEND
Position */

#define SCB_SCR_SEVONPEND_Msk       (1UL << SCB_SCR_SEVONPEND_Pos)    /*!< SCB SCR:
SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos        2                  /*!< SCB SCR: SLEEPDEEP Position
*/

#define SCB_SCR_SLEEPDEEP_Msk       (1UL << SCB_SCR_SLEEPDEEP_Pos)    /*!< SCB SCR:
SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos      1                  /*!< SCB SCR: SLEEPONEXIT
Position */

#define SCB_SCR_SLEEPONEXIT_Msk     (1UL << SCB_SCR_SLEEPONEXIT_Pos)    /*!< SCB SCR:
SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos         9                  /*!< SCB CCR: STKALIGN Position */

#define SCB_CCR_STKALIGN_Msk        (1UL << SCB_CCR_STKALIGN_Pos)    /*!< SCB CCR:
STKALIGN Mask */

```

```

#define SCB_CCR_UNALIGN_TRP_Pos      3                  /*!< SCB CCR: UNALIGN_TRP
Position */

#define SCB_CCR_UNALIGN_TRP_Msk     (1UL << SCB_CCR_UNALIGN_TRP_Pos)    /*!< SCB
CCR: UNALIGN_TRP Mask */

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_SVCALLPENDED_Pos   15                 /*!< SCB SHCSR:
SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk   (1UL << SCB_SHCSR_SVCALLPENDED_Pos)    /*!< SCB
SHCSR: SVCALLPENDED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SysTick  System Tick Timer (SysTick)
\brief Type definitions for the System Timer Registers.

@{
*/

/** \brief Structure type to access the System Timer (SysTick).
*/
typedef struct
{
    __IO uint32_t CTRL;          /*!< Offset: 0x000 (R/W) SysTick Control and Status Register */
    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) SysTick Reload Value Register */
    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) SysTick Current Value Register */
    __I uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) SysTick Calibration Register */
} SysTick_Type;

```

```

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos      16          /*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk     (1UL << SysTick_CTRL_COUNTFLAG_Pos)      /*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos     2           /*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk    (1UL << SysTick_CTRL_CLKSOURCE_Pos)      /*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos      1           /*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk     (1UL << SysTick_CTRL_TICKINT_Pos)      /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos       0           /*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk     (1UL /*<< SysTick_CTRL_ENABLE_Pos*/)      /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos      0           /*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk     (0xFFFFFFFFUL /*<< SysTick_LOAD_RELOAD_Pos*/)  /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos      0           /*!< SysTick VAL: CURRENT Position */
#define SysTick_VAL_CURRENT_Msk     (0xFFFFFFFFUL /*<< SysTick_VAL_CURRENT_Pos*/)  /*!< SysTick VAL: CURRENT Mask */

```

```

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos      31           /*!< SysTick CALIB: NOREF Position */
/*/

#define SysTick_CALIB_NOREF_Msk     (1UL << SysTick_CALIB_NOREF_Pos)      /*!< SysTick
CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos      30           /*!< SysTick CALIB: SKEW Position */
/*/

#define SysTick_CALIB_SKEW_Msk     (1UL << SysTick_CALIB_SKEW_Pos)      /*!< SysTick CALIB:
SKEW Mask */

#define SysTick_CALIB_TENMS_Pos     0            /*!< SysTick CALIB: TENMS Position */
/*/

#define SysTick_CALIB_TENMS_Msk    (0xFFFFFUL /*<< SysTick_CALIB_TENMS_Pos*/) /*!<
SysTick CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

```

```

#if (__MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUs Memory Protection Unit (MPU)
\brief Type definitions for the Memory Protection Unit (MPU)

@{
*/

```

```

/** \brief Structure type to access the Memory Protection Unit (MPU).

*/
typedef struct
{
    __I uint32_t TYPE;          /*!< Offset: 0x000 (R/ ) MPU Type Register      */
    __IO uint32_t CTRL;         /*!< Offset: 0x004 (R/W) MPU Control Register      */

```

```

__IO uint32_t RNR;           /*!< Offset: 0x008 (R/W) MPU Region RNRber Register      */
__IO uint32_t RBAR;          /*!< Offset: 0x00C (R/W) MPU Region Base Address Register    */
__IO uint32_t RASR;          /*!< Offset: 0x010 (R/W) MPU Region Attribute and Size Register */
*/
} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos     16          /*!< MPU TYPE: IREGION Position */
#define MPU_TYPE_IREGION_Msk     (0xFFUL << MPU_TYPE_IREGION_Pos) /*!< MPU TYPE: IREGION Mask */

#define MPU_TYPE_DREGION_Pos     8           /*!< MPU TYPE: DREGION Position */
#define MPU_TYPE_DREGION_Msk     (0xFFUL << MPU_TYPE_DREGION_Pos) /*!< MPU TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos    0           /*!< MPU TYPE: SEPARATE Position */
#define MPU_TYPE_SEPARATE_Msk    (1UL /*<< MPU_TYPE_SEPARATE_Pos*/) /*!< MPU TYPE: SEPARATE Mask */

/* MPU Control Register */

#define MPU_CTRL_PRIVDEFENA_Pos   2           /*!< MPU CTRL: PRIVDEFENA Position */
#define MPU_CTRL_PRIVDEFENA_Msk  (1UL << MPU_CTRL_PRIVDEFENA_Pos) /*!< MPU CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos    1           /*!< MPU CTRL: HFNMIENA Position */
#define MPU_CTRL_HFNMIENA_Msk   (1UL << MPU_CTRL_HFNMIENA_Pos) /*!< MPU CTRL: HFNMIENA Mask */

```

```

#define MPU_CTRL_ENABLE_Pos      0           /*!< MPU CTRL: ENABLE Position */
#define MPU_CTRL_ENABLE_Msk     (1UL /*<< MPU_CTRL_ENABLE_Pos*/)    /*!< MPU
CTRL: ENABLE Mask */

/* MPU Region Number Register */
#define MPU_RNR_REGION_Pos      0           /*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk     (0xFFUL /*<< MPU_RNR_REGION_Pos*/)    /*!< MPU
RNR: REGION Mask */

/* MPU Region Base Address Register */
#define MPU_RBAR_ADDR_Pos       8           /*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk     (0xFFFFFUL << MPU_RBAR_ADDR_Pos)    /*!< MPU
RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos       4           /*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk     (1UL << MPU_RBAR_VALID_Pos)    /*!< MPU RBAR:
VALID Mask */

#define MPU_RBAR_REGION_Pos      0           /*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk     (0xFUL /*<< MPU_RBAR_REGION_Pos*/)    /*!< MPU
RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */
#define MPU_RASR_ATTRS_Pos       16          /*!< MPU RASR: MPU Region
Attribute field Position */
#define MPU_RASR_ATTRS_Msk     (0xFFFFUL << MPU_RASR_ATTRS_Pos)    /*!< MPU RASR:
MPU Region Attribute field Mask */

#define MPU_RASR_XN_Pos         28          /*!< MPU RASR: ATTRS.XN Position
*/

```

#define MPU_RASR_XN_Msk	(1UL << MPU_RASR_XN_Pos)	/*!< MPU RASR: ATTRS.XN Mask */
#define MPU_RASR_AP_Pos	24	/*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk	(0x7UL << MPU_RASR_AP_Pos)	/*!< MPU RASR: ATTRS.AP Mask */
#define MPU_RASR_TEX_Pos	19	/*!< MPU RASR: ATTRS.TEX Position */
#define MPU_RASR_TEX_Msk	(0x7UL << MPU_RASR_TEX_Pos)	/*!< MPU RASR: ATTRS.TEX Mask */
#define MPU_RASR_S_Pos	18	/*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk	(1UL << MPU_RASR_S_Pos)	/*!< MPU RASR: ATTRS.S Mask */
#define MPU_RASR_C_Pos	17	/*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk	(1UL << MPU_RASR_C_Pos)	/*!< MPU RASR: ATTRS.C Mask */
#define MPU_RASR_B_Pos	16	/*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk	(1UL << MPU_RASR_B_Pos)	/*!< MPU RASR: ATTRS.B Mask */
#define MPU_RASR_SRD_Pos	8	/*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk	(0xFFUL << MPU_RASR_SRD_Pos)	/*!< MPU RASR: Sub-Region Disable Mask */
#define MPU_RASR_SIZE_Pos	1	/*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk	(0x1FUL << MPU_RASR_SIZE_Pos)	/*!< MPU RASR: Region Size Field Mask */

```
#define MPU_RASR_ENABLE_Pos      0           /*!< MPU RASR: Region enable bit Position */

#define MPU_RASR_ENABLE_Msk      (1UL/*<< MPU_RASR_ENABLE_Pos*/)    /*!< MPU RASR: Region enable bit Disable Mask */

/*@} end of group CMSIS_MPU */

#endif
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug   Core Debug Registers (CoreDebug)
\brief   Cortex-M0+ Core Debug Registers (DCB registers, SHCSR, and DFSR)
are only accessible over DAP and not via processor. Therefore
they are not covered by the Cortex-M0 header file.
```

```
@{
*/
/*@} end of group CMSIS_CoreDebug */
```

```
/** \ingroup CMSIS_core_register
\defgroup CMSIS_core_base   Core Definitions
\brief   Definitions for base addresses, unions, and structures.
@{
*/

```

```
/* Memory mapping of Cortex-M0+ Hardware */

#define SCS_BASE      (0xE000E000UL)        /*!< System Control Space Base Address */
#define SysTick_BASE  (SCS_BASE + 0x0010UL)    /*!< SysTick Base Address */
```

```

#define NVIC_BASE      (SCS_BASE + 0x0100UL)           /*!< NVIC Base Address      */
#define SCB_BASE       (SCS_BASE + 0x0D00UL)           /*!< System Control Block Base Address */

#define SCB          ((SCB_Type    *) SCB_BASE)        /*!< SCB configuration struct      */
#define SysTick      ((SysTick_Type *) SysTick_BASE)  /*!< SysTick configuration struct      */
#define NVIC         ((NVIC_Type   *) NVIC_BASE)        /*!< NVIC configuration struct      */

#if (__MPU_PRESENT == 1)

#define MPU_BASE      (SCS_BASE + 0x0D90UL)           /*!< Memory Protection Unit      */
#define MPU          ((MPU_Type    *) MPU_BASE)        /*!< Memory Protection Unit      */

#endif

/*@} */

```

```

*****
*      Hardware Abstraction Layer
Core Function Interface contains:
- Core NVIC Functions
- Core SysTick Functions
- Core Register Access Functions
*****
/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/

```

```

/* ##### NVIC functions #####

```

```

/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_NVICFunctions NVIC Functions
 \brief Functions that manage interrupts and exceptions via the NVIC.

 @{
 */

/* Interrupt Priorities are WORD accessible only under ARMv6M */
/* The following MACROS handle generation of the register offset and byte masks */

#define _BIT_SHIFT(IRQn)    ( (((uint32_t)(int32_t)(IRQn))      ) & 0x03UL) * 8UL)
#define _SHP_IDX(IRQn)      ( (((((uint32_t)(int32_t)(IRQn)) & 0x0FUL)-8UL) >> 2UL) )
#define _IP_IDX(IRQn)       ( (((uint32_t)(int32_t)(IRQn))           >> 2UL) )

```

/** \brief Enable External Interrupt

The function enables a device-specific interrupt in the NVIC interrupt controller.

```

\param [in] IRQn External interrupt number. Value cannot be negative.

*/
__STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
    NVIC->ISER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}

```

/** \brief Disable External Interrupt

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]    IRQn  External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Get Pending Interrupt
```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

```
\param [in]    IRQn  Interrupt number.  
  
\return      0  Interrupt status is not pending.  
\return      1  Interrupt status is pending.  
*/  
STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)  
{  
    return((uint32_t)((NVIC->ISPR[0] & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL :  
        0UL);  
}
```

```
/** \brief Set Pending Interrupt
```

The function sets the pending bit of an external interrupt.

```
\param [in]    IRQn  Interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Clear Pending Interrupt
```

The function clears the pending bit of an external interrupt.

```
\param [in]    IRQn  External interrupt number. Value cannot be negative.  
*/  
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ICPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));  
}
```

```
/** \brief Set Interrupt Priority
```

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

```
\param [in]    IRQn  Interrupt number.  
\param [in]    priority  Priority to set.  
*/
```

```

__STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
    if((int32_t)(IRQn) < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = ((uint32_t)(SCB->SHP[_SHP_IDX(IRQn)] & ~(0xFFUL << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) << _BIT_SHIFT(IRQn)));
    }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = ((uint32_t)(NVIC->IP[_IP_IDX(IRQn)] & ~(0xFFUL << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) << _BIT_SHIFT(IRQn)));
    }
}

```

/** \brief Get Interrupt Priority

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

\param [in] IRQn Interrupt number.

\return Interrupt Priority. Value is aligned automatically to the implemented priority bits of the microcontroller.

*/

```

__STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{

```

if((int32_t)(IRQn) < 0) {

```

    return((uint32_t)((SCB->SHP[_SHP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & (uint32_t)0xFFUL) >> (8 -
__NVIC_PRIO_BITS));
}

else {

    return((uint32_t)((NVIC->IP[_IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & (uint32_t)0xFFUL) >> (8 -
__NVIC_PRIO_BITS));
}

}

```

/** \brief System Reset

The function initiates a system reset request to reset the MCU.

```

*/
__STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                /* Ensure all outstanding memory accesses included
                                              buffered write are completed before reset */

    SCB->AIRCR = ((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                    SCB_AIRCR_SYSRESETREQ_Msk);

    __DSB();                                /* Ensure completion of memory access */

    while(1) { __NOP(); }                   /* wait until reset */
}

```

/*@} end of CMSIS_Core_NVICFunctions */

```

/* ##### SysTick function
##### */

```

```

/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_Core_SysTickFunctions SysTick Functions
 \brief Functions that configure the System.

 @{
 */

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig 0

/** \brief System Tick Configuration

The function initializes the System Timer and its interrupt, and starts the System Tick Timer.
Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.
\return 1 Function failed.

\note When the variable <b>__Vendor_SysTickConfig</b> is set to 1, then the
function <b>SysTick_Config</b> is not included. In this case, the file <b><i>device</i>.h</b>
must contain a vendor-specific implementation of this function.

*/
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) {return (1UL); /* Reload value impossible */

    SysTick->LOAD = (uint32_t)(ticks - 1UL); /* set reload register */
}

```

```

    NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
}

SysTick->VAL  = 0UL;                      /* Load the SysTick Counter Value */

SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
                  SysTick_CTRL_TICKINT_Msk |
                  SysTick_CTRL_ENABLE_Msk;      /* Enable SysTick IRQ and SysTick Timer */

return (0UL);                            /* Function successful */

}

#endif

/*}@} end of CMSIS_Core_SysTickFunctions */

#endif /* __cplusplus

}

#endif /* __CORE_CM0PLUS_H_DEPENDANT */

#endif /* __CMSIS_GENERIC */

#ifndef INCLUDES_PROJECT2_H_
#define INCLUDES_PROJECT2_H_

void project2();

#endif /* INCLUDES_PROJECT2_H_ */

```

```
#ifndef INCLUDES_CONVERSION_H_
#define INCLUDES_CONVERSION_H_
/*********************  

 * * Authors: Harsimransingh Bindra, Smitesh Modak  

 * * Date Edited: 10/01/2017  

 * *  

 * * File: conversion.h  

 * *  

 * * Description: Header file for data manipulation  

 * * -my_itoa  

 * * -my_atoi  

 * * -dump_memory  

 * * -big_to_little32  

 * * -little_to_big32  

 * *  

 * *  

 * *  

 * ****header file for function called "conversion"*/
```

```
typedef enum conversion_status{
    Invalid_pointer,
    Pass,
    Fail
}conversion_status_t;
```

```
uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base);
/**
```

```

* Function converting integral values to strings
*
* This function takes an integer value as input and converts it into a string and handles both the signed
and unsigned data.

* It handles number from base 2 to base 16
*
* returns uint8_t*/
int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base);

/**Function converting string ascii inputs to integers
*
* This function takes a string as input and converts it into a integer value and handles both the signed
and unsigned data.

* It handles number from base 2 to base 16
*
* returns uint8_t*/
int8_t big_to_little32(uint32_t *data, uint32_t length);

/**
* Changing endianness from big endian to little endian
*
* This function takes data which is saved in big endian format and changes the endianness to little
endianness.

* This function is used when the data needs to be transferred to different platforms with different
endianness
*
* returns int8_t*/
int8_t little_to_big32(uint32_t *data, uint32_t length);

/**
*Changing endianness from little endian to big endian
*

```

* This function takes data which is saved in little endian format and changes the endianness to big endian.

* This function is used when the data needs to be transferred to different platforms with different endianness

*

* returns int8_t*/

#endif /* INCLUDES_CONVERSION_H */

/*

** #####

** Processors: MKL25Z128FM4

** MKL25Z128FT4

** MKL25Z128LH4

** MKL25Z128VLK4

**

** Compilers: Keil ARM C/C++ Compiler

** Freescale C/C++ for Embedded ARM

** GNU C Compiler

** GNU C Compiler - CodeSourcery Sourcery G++

** IAR ANSI C/C++ Compiler for ARM

**

** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012

** Version: rev. 2.5, 2015-02-19

** Build: b150220

**

** Abstract:

** Provides a system configuration function and a global variable that

** contains the system frequency. It configures the device and initializes

** the oscillator (PLL) that is part of the microcontroller device.

**

** Copyright (c) 2015 Freescale Semiconductor, Inc.

** All rights reserved.

**

** Redistribution and use in source and binary forms, with or without modification,

** are permitted provided that the following conditions are met:

**

** o Redistributions of source code must retain the above copyright notice, this list

** of conditions and the following disclaimer.

**

** o 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.

**

** o Neither the name of Freescale Semiconductor, Inc. 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.

**

** http: www.freescale.com
** mail: support@freescale.com
**
** Revisions:
** - rev. 1.0 (2012-06-13)
** Initial version.
** - rev. 1.1 (2012-06-21)
** Update according to reference manual rev. 1.
** - rev. 1.2 (2012-08-01)
** Device type UARTLP changed to UART0.
** - rev. 1.3 (2012-10-04)
** Update according to reference manual rev. 3.
** - rev. 1.4 (2012-11-22)
** MCG module - bit LOLS in MCG_S register renamed to LOLS0.
** NV registers - bit EZPORT_DIS in NV_FOPT register removed.
** - rev. 1.5 (2013-04-05)
** Changed start of doxygen comment.
** - rev. 2.0 (2013-10-29)
** Register accessor macros added to the memory map.
** Symbols for Processor Expert memory map compatibility added to the memory map.
** Startup file for gcc has been updated according to CMSIS 3.2.
** System initialization updated.
** - rev. 2.1 (2014-07-16)
** Module access macro module_BASES replaced by module_BASE_PTRS.
** System initialization and startup updated.
** - rev. 2.2 (2014-08-22)
** System initialization updated - default clock config changed.
** - rev. 2.3 (2014-08-28)
** Update of startup files - possibility to override DefaultISR added.

```
** - rev. 2.4 (2014-10-14)
**     Interrupt INT_LPTimer renamed to INT_LPTMR0.
** - rev. 2.5 (2015-02-19)
**     Renamed interrupt vector LLW to LLWU.
**
** #####
*/

```

```
/*!
* @file MKL25Z4
* @version 2.5
* @date 2015-02-19
* @brief Device specific configuration file for MKL25Z4 (header file)
*
* Provides a system configuration function and a global variable that contains
* the system frequency. It configures the device and initializes the oscillator
* (PLL) that is part of the microcontroller device.
*/

```

```
#ifndef SYSTEM_MKL25Z4_H_
#define SYSTEM_MKL25Z4_H_          /**< Symbol preventing repeated inclusion */

#ifndef __cplusplus
extern "C" {
#endif

#include <stdint.h>
```

```
#ifndef DISABLE_WDOG
#define DISABLE_WDOG      1
#endif

/* MCG mode constants */

#define MCG_MODE_FEI      0U
#define MCG_MODE_FBI      1U
#define MCG_MODE_BLPI     2U
#define MCG_MODE_FEE      3U
#define MCG_MODE_FBE      4U
#define MCG_MODE_BLPE     5U
#define MCG_MODE_PBE      6U
#define MCG_MODE_PEE      7U

/* Predefined clock setups

0 ... Default part configuration

    Multipurpose Clock Generator (MCG) in FEI mode.

    Reference clock source for MCG module: Slow internal reference clock
    Core clock = 20.97152MHz
    Bus clock = 20.97152MHz

1 ... Maximum achievable clock frequency configuration

    Multipurpose Clock Generator (MCG) in PEE mode.

    Reference clock source for MCG module: System oscillator reference clock
    Core clock = 48MHz
    Bus clock = 24MHz
```

2 ... Chip internally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPI mode.

Reference clock source for MCG module: Fast internal reference clock

Core clock = 4MHz

Bus clock = 0.8MHz

3 ... Chip externally clocked, ready for Very Low Power Run mode

Multipurpose Clock Generator (MCG) in BLPE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 4MHz

Bus clock = 1MHz

4 ... USB clock setup

Multipurpose Clock Generator (MCG) in PEE mode.

Reference clock source for MCG module: System oscillator reference clock

Core clock = 48MHz

Bus clock = 24MHz

*/

/* Define clock source values */

```
#define CPU_XTAL_CLK_HZ      8000000u      /* Value of the external crystal or oscillator clock  
frequency in Hz */
```

```
#define CPU_INT_SLOW_CLK_HZ   32768u       /* Value of the slow internal oscillator clock  
frequency in Hz */
```

```
#define CPU_INT_FAST_CLK_HZ   4000000u     /* Value of the fast internal oscillator clock  
frequency in Hz */
```

/* RTC oscillator setting */

/* Low power mode enable */

/* SMC_PMPROT: AVLP=1,ALLS=1,AVLLS=1 */

```

#define SYSTEM_SMC_PMPROT_VALUE      0x2AU      /* SMC_PMPROT */

/* Internal reference clock trim */

/* #undef SLOW_TRIM_ADDRESS */          /* Slow oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef SLOW_FINE_TRIM_ADDRESS */     /* Slow oscillator not trimmed. Commented out
for MISRA compliance. */

/* #undef FAST_TRIM_ADDRESS */         /* Fast oscillator not trimmed. Commented out for
MISRA compliance. */

/* #undef FAST_FINE_TRIM_ADDRESS */    /* Fast oscillator not trimmed. Commented out
for MISRA compliance. */

#endif CLOCK_SETUP

#if (CLOCK_SETUP == 0)

#define DEFAULT_SYSTEM_CLOCK      20971520u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_FEI /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE      0x06U      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSS0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE      0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRSS=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE      0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE      0x00U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE      0x00U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

```

```

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE 0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=0 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x00U      /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE 0x01000000U /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 1)

#define DEFAULT_SYSTEM_CLOCK 48000000u /* Default System clock value */

#define MCG_MODE          MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE 0x1AU /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSD0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE 0x24U /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE 0x00U /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSrv=0,FCRDIV=0,LOCs0=0 */

#define SYSTEM_MCG_SC_VALUE 0x00U /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=1 */

#define SYSTEM_MCG_C5_VALUE 0x01U /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE 0x40U /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE 0x80U /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE 0x00U /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x10010000U /* SIM_CLKDIV1 */

```

```

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U    /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 2)

#define DEFAULT_SYSTEM_CLOCK     4000000u      /* Default System clock value */

#define MCG_MODE                 MCG_MODE_BLPI /* Clock generator mode */

/* MCG_C1: CLKS=1,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE     0x46U        /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFS0=1,LP=1,IRCS=1 */

#define SYSTEM_MCG_C2_VALUE     0x27U        /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DR=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE     0x00U        /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE     0x00U        /* MCG_SC */

/* MCG_C5: PLLCLKENO=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE     0x00U        /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE     0x00U        /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE    0x80U        /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE 0x00U        /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=4 */

#define SYSTEM_SIM_CLKDIV1_VALUE 0x00040000U   /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x02000000U    /* SIM_SOPT2 */

```

```

#elif (CLOCK_SETUP == 3)

#define DEFAULT_SYSTEM_CLOCK      4000000u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_BLPE /* Clock generator mode */

/* MCG_C1: CLKS=2,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

#define SYSTEM_MCG_C1_VALUE      0x9AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSO=1,LP=1,IRCS=1 */

#define SYSTEM_MCG_C2_VALUE      0x27U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=1,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE      0x02U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTENO=0,PRDIV0=0 */

#define SYSTEM_MCG_C5_VALUE      0x00U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE      0x00U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=3 */

#define SYSTEM_SIM_CLKDIV1_VALUE  0x10030000U    /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U    /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x02000000U    /* SIM_SOPT2 */

#elif (CLOCK_SETUP == 4)

#define DEFAULT_SYSTEM_CLOCK      48000000u      /* Default System clock value */

#define MCG_MODE                  MCG_MODE_PEE /* Clock generator mode */

/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */

```

```

#define SYSTEM_MCG_C1_VALUE      0x1AU      /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREF0=1,LP=0,IRCS=0 */

#define SYSTEM_MCG_C2_VALUE      0x24U      /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DR0=0,FCTRIM=0,SCFTRIM=0 */

#define SYSTEM_MCG_C4_VALUE      0x00U      /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCS0=0 */

#define SYSTEM_MCG_SC_VALUE      0x00U      /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=1 */

#define SYSTEM_MCG_C5_VALUE      0x01U      /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */

#define SYSTEM_MCG_C6_VALUE      0x40U      /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */

#define SYSTEM_OSC0_CR_VALUE     0x80U      /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */

#define SYSTEM_SMC_PMCTRL_VALUE   0x00U      /* SMC_PMCTRL */

/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */

#define SYSTEM_SIM_CLKDIV1_VALUE  0x10010000U /* SIM_CLKDIV1 */

/* SIM_SOPT1: USBREGEN=0,USBSSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */

#define SYSTEM_SIM_SOPT1_VALUE    0x000C0000U /* SIM_SOPT1 */

/* SIM_SOPT2: UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */

#define SYSTEM_SIM_SOPT2_VALUE    0x01010000U /* SIM_SOPT2 */

#endif

#ifndef __ASSEMBLY__
#define SYSTEM_CLOCK_DEFAULT      20971520U /* Default System clock value */
#endif

/** 
 * @brief System clock frequency (core clock)

```

```
*  
* The system clock frequency supplied to the SysTick timer and the processor  
* core clock. This variable can be used by the user application to setup the  
* SysTick timer or configure other parameters. It may also be used by debugger to  
* query the frequency of the debug timer or configure the trace clock speed  
* SystemCoreClock is initialized with a correct predefined value.  
*/
```

```
extern uint32_t SystemCoreClock;
```

```
/**  
* @brief Setup the microcontroller system.  
*  
* Typically this function configures the oscillator (PLL) that is part of the  
* microcontroller device. For systems with variable clock speed it also updates  
* the variable SystemCoreClock. SystemInit is called from startup_device file.  
*/
```

```
void SystemInit (void);
```

```
/**  
* @brief Updates the SystemCoreClock variable.  
*  
* It must be called whenever the core clock is changed during program  
* execution. SystemCoreClockUpdate() evaluates the clock register settings and calculates  
* the current core clock.  
*/
```

```
void SystemCoreClockUpdate (void);
```

```
#ifdef __cplusplus
```

```
}
```

```

#endif

#endif /* #if !defined(SYSTEM_MKL25Z4_H_) */

//****************************************************************************
** Authors : Harsimransingh Bindra, Smitesh Modak
** Date : 10/28/2017
**
** File : circbuff.c
**
** Description : Source file for Circular buffer functions
**
** CB_status CB_init(CB_t *cbuf,uint8_t length)
**
** CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val)
**
** CB_status CB_buffer_remove_item(CB_t *cbuf)
**
** CB_status CB_is_full(CB_t *cbuf)
**
** CB_status CB_is_empty(CB_t *cbuf)
**
** CB_status CB_peek(CB_t *cbuf, uint8_t position)
**
** CB_status CB_destroy(CB_t *cbuf)
**
** */

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include "circbuf.h"

/*This function is used to initialize circular buffer, to make head,tail
 * and buffer point at first location of buffer and initialize count to 0*/
CB_status CB_init(CB_t *cbuf,uint8_t length)
{
    cbuf->buffer = NULL;
    cbuf->buffer = malloc(sizeof(uint8_t)*length);
}

```

```

cbuf->length = length;
cbuf->head = cbuf->buffer;
cbuf->tail = cbuf->buffer;
cbuf->count = 0;
cbuf->peek = 0;
return No_error;
}

/*This function is used to add element to circular buffer*/
CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val)
{ if(!cbuf->tail)
{ /*Return null pointer error if Cbuf doesn't point to a valid location*/
    return Null_pointer;
}
if(cbuf->count == cbuf->length)
{
/*Condition to check if buffer is not full*/
    return Full;
}
*(cbuf->tail) = val;
/*Put the data at location pointed by head*/
if(cbuf->tail==cbuf->buffer+(cbuf->length-1))
{
/*Wrap around, if tail is pointing to last location of buffer*/
    cbuf->tail = cbuf->buffer;
}
else
{
    cbuf->tail++;
}

```

```

    }

    cbuf->count=cbuf->count+1;

/*Return No_error after adding element in buffer*/

return No_error;

}

/*This function is used to remove elements from circular buffer*/

CB_status CB_buffer_remove_item(CB_t *cbuf)

{

if(!cbuf->tail)

{ /*Return null pointer error if Cbuf doesn't point to a valid location*/

    return Null_pointer;

}

if(cbuf->count==0)

{

/*Count value checks if buffer is empty*/

    return Empty;

}

/*After removing the element from buffer, increment tail pointer
 *
 *      and decrement count value*/

*(cbuf->head)=0;

if(cbuf->head == cbuf->buffer+cbuf->length-sizeof(char))

{

/*Wrap around if head points at last location*/

    cbuf->head = cbuf->buffer;

}

else

{

```

```

cbuf->head++;
}

cbuf->count = cbuf->count-1;

cbuf->status=No_error;

return No_error;

}

/*This function checks if buffer is full*/

CB_status CB_is_full(CB_t *cbuf)

{

if(cbuf == NULL)

{

    return Null_pointer;

}

if(cbuf->count == cbuf->length)

{

    //If count value is equal to length, return circular buffer is full

    return Full;

}

else

/*If count is less than length, return circular buffer is available*/

    return Available;

}

/*This function checks if buffer is empty*/

CB_status CB_is_empty(CB_t *cbuf)

{

if(cbuf == NULL)

{

```

```

/*return null pointer error,if Cbuf doesn't point to a valid location*/
return Null_pointer;
}

if(cbuf->count == 0)

{
    /*If Count=0, return circular buffer is empty*/
    return Empty;
}

else

/*If count!=0 return buffer is available*/
return Available;

}

```

```

/*This function is used to check the value present at a particular
 * location of circular buffer*/

CB_status CB_peek(CB_t *cbuf, uint8_t position)

{
    if(cbuf == NULL)

    {
        /*Return null pointer error, if CircbuffTR doesn't point to a valid location*/
        return Null_pointer;
    }

    else if(position > cbuf->length)

    {
        return Wrong_memory_access;
    }

    else if(position > cbuf->count)

    {
        return Empty;
    }
}

```

```

    }

else

{

volatile uint8_t *pos = cbuf->head;

while(position)

{

if(pos == cbuf->buffer+cbuf->length-sizeof(uint8_t))

{

    pos = cbuf->buffer;

}

else

{

    pos = pos+1;

}

position--;

}

cbuf->peek = *pos;

}

return No_error;
}

```

```

/*This function is used to free the heap memory allocate to buffer*/

CB_status CB_destroy(CB_t *cbuf)

{

free((uint8_t *)cbuf->buffer);

free(cbuf);

/*After destroying memory return no error*/

return No_error;
}

```

```
#define LargestIntegralType unsigned long long

void _assert_true(const LargestIntegralType result,
                 const char* const expression,
                 const char * const file, const int line)
{
    __coverity_panic__();
}

void _assert_int_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line)
{
    __coverity_panic__();
}

void _assert_int_not_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line)
{
    __coverity_panic__();
}

void _assert_return_code(const LargestIntegralType result,
                        size_t rlen,
                        const LargestIntegralType error,
                        const char * const expression,
                        const char * const file,
                        const int line)
```

```
{  
    __coverity_panic__();  
}  
  
void _assert_string_equal(const char * const a, const char * const b,  
                        const char * const file, const int line)  
{  
    __coverity_panic__();  
}  
  
void _assert_string_not_equal(const char * const a, const char * const b,  
                            const char *file, const int line)  
{  
    __coverity_panic__();  
}  
  
void _assert_memory_equal(const void * const a, const void * const b,  
                        const size_t size, const char* const file,  
                        const int line)  
{  
    __coverity_panic__();  
}  
  
void _assert_memory_not_equal(const void * const a, const void * const b,  
                            const size_t size, const char* const file,  
                            const int line)  
{  
    __coverity_panic__();  
}
```

```
void _assert_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line)
{
    __coverity_panic__();
}

void _assert_not_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line)
{
    __coverity_panic__();
}

void _assert_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line)
{
    __coverity_panic__();
}

void _assert_not_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line)
{
    __coverity_panic__();
}
```

```
/* Functions to help coverity do static analysis on cmocka */

void exit_test(const int quit_application)

{

    __coverity_panic__();

}

#include "config.h"

#include <stdarg.h>

#include <stddef.h>

#include <setjmp.h>

#include <cmocka.h>

#include <cmocka_private.h>

#include <stdlib.h>

int mock_function(void);

void mock_function_call_times(size_t times, int expectedValue);

int mock_function(void)

{

    return (int) mock();

}

void mock_function_call_times(size_t times, int expectedValue)

{

    size_t i;

    for (i = 0; i < times; ++i)

    {

        assert_int_equal(expectedValue, mock_function());

    }

}
```

```
    }

}

static void test_will_return_maybe_for_no_calls(void **state)
{
    (void) state;

    will_return_maybe(mock_function, 32);
}
```

```
static void test_will_return_maybe_for_one_mock_call(void **state)
{
    int value;

    (void) state;

    value = rand();
    will_return_maybe(mock_function, value);
    mock_function_call_times(1u, value);
}
```

```
static void test_will_return_maybe_for_more_than_one_call(void **state)
{
    int value;
    size_t numberOfCalls;
    (void)state;

    value = rand();
    numberOfCalls = (size_t) ((rand()) % 20 + 2);
```

```
    will_return_maybe(mock_function, value);
    mock_function_call_times(numberOfCalls, value);
}

int main(int argc, char **argv) {
    const struct CMUnitTest alloc_tests[] = {
        cmocka_unit_test(test_will_return_maybe_for_no_calls)
        ,cmocka_unit_test(test_will_return_maybe_for_one_mock_call)
        ,cmocka_unit_test(test_will_return_maybe_for_more_than_one_call)
    };

    (void)argc;
    (void)argv;

    return cmocka_run_group_tests(alloc_tests, NULL, NULL);
}

#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>

#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#ifndef HAVE_UNISTD_H
#include <unistd.h>
```

```
#endif

#include <fcntl.h>

/***********************/

*** assert_return_code
***********************/

static void test_assert_return_code(void **state)
{
    struct stat sb;
    int rc;

    (void)state; /* unused */

    rc = stat(".", &sb);
    assert_return_code(rc, 0);

#ifndef _MSC_VER
    assert_true(S_ISDIR(sb.st_mode));
#endif

}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_assert_return_code),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/* Use the unit test allocators */
```

```
#define UNIT_TESTING 1

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

static int group_setup_failing(void **state)
{
    (void) state; /* unused */
    return 1; /* To indicate the failure */
}

static void test_true(void **state)
{
    (void) state; /* unused */
    assert_true(1);
}

static void test_false(void **state)
{
    (void) state; /* unused */
    assert_false(0);
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_true),
        cmocka_unit_test(test_false),
    }
}
```

```
};

return cmocka_run_group_tests(tests, group_setup_failing, NULL);
}

#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>

#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#ifndef HAVE_UNISTD_H
#include <unistd.h>
#endif
#ifndef HAVE_IO_H
#include <io.h>
#endif
#include <fcntl.h>

*****
*** assert_return_code
*****/

static void test_assert_return_code_fail(void **state)
{
    int fd;
```

```
(void)state; /* unused */

fd = open("this_file_doesnt_exist.cmocka", 0);
assert_return_code(fd, errno);

if (fd >= 0) {
    close(fd);
}

}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_assert_return_code_fail),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/* Use the unit test allocators */

#define UNIT_TESTING 1

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

static int group_setup(void **state)
{
    int *answer = malloc(sizeof(int));
```

```
assert_non_null(answer);

*answer = 42;

*state = answer;

return 0;
}

static int group_teardown(void **state)
{
    int *answer = (int *)*state;

    free(answer);

    return 0;
}

static void test_value_equal(void **state)
{
    int a = *((int *)*state);

    assert_int_equal(a, 42);
}

static void test_value_range(void **state)
{
    int a = *((int *)*state);

    assert_in_range(a, 0, 100);
}
```

```
int main(void) {
    int prestate = 1337;
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_value_equal),
        cmocka_unit_test(test_value_range),
        cmocka_unit_test_prestate(test_value_equal, &prestate),
    };

    return cmocka_run_group_tests(tests, group_setup, group_teardown);
}

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

#include <stdlib.h>

struct test_segv {
    int x;
    int y;
};

static void test_segfault_recovery(void **state)
{
    struct test_segv *s = NULL;
    (void) state; /* unused */

    s->x = 1;
```

```
}
```

```
int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_segfault_recovery),
        cmocka_unit_test(test_segfault_recovery),
        cmocka_unit_test(test_segfault_recovery),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */


```

```
#include <stdarg.h>
```

```
#include <stddef.h>
```

```
#include <setjmp.h>
#include <cmocka.h>

/* A test case that does check if an int is equal. */
static void test_check_skip(void **state) {
    (void)state; /* unused */

    skip();

    assert_true(0);
}
```

```
int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_check_skip),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}
```

```
#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>
```

```
static void mock_test_a_called(void)
{
    function_called();
}

static void mock_test_b_called(void)
{
    function_called();
}

static void mock_test_c_called(void)
{
    function_called();
}

static void test_does_fail_for_unexpected_call(void **state)
{
    (void)state;
    expect_function_call(mock_test_a_called);
    expect_function_call(mock_test_a_called);

    mock_test_a_called();
    mock_test_a_called();
    mock_test_a_called();

}

static void test_does_fail_for_unmade_expected_call(void **state)
{
    (void)state;
```

```
expect_function_call(mock_test_a_called);
expect_function_call(mock_test_a_called);

mock_test_a_called();
}

static void test_ordering_fails_out_of_order(void **state)
{
    (void)state;
    expect_function_call(mock_test_a_called);
    expect_function_call(mock_test_b_called);
    expect_function_call(mock_test_a_called);

    mock_test_b_called();
}

static void test_ordering_fails_out_of_order_for_at_least_once_calls(void **state)
{
    (void)state;
    expect_function_call_any(mock_test_a_called);
    ignore_function_calls(mock_test_b_called);

    mock_test_b_called();
    mock_test_c_called();
}

/* Primarily used to test error message */

static void test_fails_out_of_order_if_no_calls_found_on_any(void **state)
{
```

```

(void)state;
expect_function_call_any(mock_test_a_called);
ignore_function_calls(mock_test_b_called);

mock_test_a_called();
mock_test_c_called();

}

static void test_fails_if_zero_count_used(void **state)
{
(void)state;
expect_function_calls(mock_test_a_called, 0);

mock_test_a_called();

}

int main(void) {
const struct CMUnitTest tests[] = {
    cmocka_unit_test(test_does_fail_for_unexpected_call)
    ,cmocka_unit_test(test_does_fail_for_unmade_expected_call)
    ,cmocka_unit_test(test_does_fail_for_unmade_expected_call)
    ,cmocka_unit_test(test_ordering_fails_out_of_order)
    ,cmocka_unit_test(test_ordering_fails_out_of_order_for_at_least_once_calls)
    ,cmocka_unit_test(test_fails_out_of_order_if_no_calls_found_on_any)
    ,cmocka_unit_test(test_fails_if_zero_count_used)
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

```

```
/*
 * Copyright 2016 David Schneider <schneidav81@gmail.com>
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 *
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
```

```
/* Use the unit test allocators */
```

```
#define UNIT_TESTING 1
```

```
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
```

```
static int setup(void **state) {
    int *answer = malloc(sizeof(int));
```

```
    assert_non_null(answer);
    *answer = 42;
```

```
*state = answer;

return 0;
}

static int teardown(void **state) {
    free(*state);

    return 0;
}

/* A test case that does nothing and succeeds.*/
static void null_test_success(void **state) {
    (void) state;
}

/* A test case that does check if an int is equal.*/
static void int_test_success(void **state) {
    int *answer = *state;

    assert_int_equal(*answer, 42);
}

int main(void) {
    const struct CMUnitTest test_group1[] = {
        cmocka_unit_test(null_test_success),
    };
}
```

```
const struct CMUnitTest test_group2[] = {
    cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),
};

int result = 0;
result += cmocka_run_group_tests(test_group1, NULL, NULL);
result += cmocka_run_group_tests(test_group2, NULL, NULL);

return result;
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <stdarg.h>
#include <stddef.h>
```

```
#include <setjmp.h>
#include <cmockery/cmockery.h>

/* A test case that does nothing and succeeds. */
static void null_test_success(void **state) {
    (void) state; /* unused */
}
```

```
int main(void) {
    const UnitTest tests[] = {
        unit_test(null_test_success),
    };
    return run_tests(tests);
}

/* Use the unit test allocators */

#define UNIT_TESTING 1
```

```
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

static int group_setup_failing(void **state)
{
    (void) state; /* unused */

    assert_int_equal(0, 1);

    return 0;
```

```
}

static void test_true(void **state)
{
    (void) state; /* unused */
    assert_true(1);
}

static void test_false(void **state)
{
    (void) state; /* unused */
    assert_false(0);
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_true),
        cmocka_unit_test(test_false),
    };

    return cmocka_run_group_tests(tests, group_setup_failing, NULL);
}

#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>
```

```
static void mock_test_a_called(void)
```

```
{
```

```
    function_called();
```

```
}
```

```
static void mock_test_b_called(void)
```

```
{
```

```
    function_called();
```

```
}
```

```
static void mock_test_c_called(void)
```

```
{
```

```
    function_called();
```

```
}
```

```
static void test_does_succeed_for_expected(void **state)
```

```
{
```

```
    (void)state;
```

```
    expect_function_call(mock_test_a_called);
```

```
    expect_function_call(mock_test_a_called);
```

```
    mock_test_a_called();
```

```
    mock_test_a_called();
```

```
}
```

```
static void test_does_succeed_for_multiple_calls(void **state)
```

```
{
```

```
(void)state;  
expect_function_call(mock_test_a_called);  
expect_function_calls(mock_test_a_called, 2);  
expect_function_call(mock_test_a_called);  
  
mock_test_a_called();  
mock_test_a_called();  
mock_test_a_called();  
mock_test_a_called();  
}  
  
static void test_ordering_does_ignore_calls(void **state)  
{  
    (void)state;  
  
    ignore_function_calls(mock_test_a_called);  
  
    mock_test_a_called();  
    mock_test_a_called();  
    mock_test_a_called();  
}  
  
static void test_ordering_does_ignore_no_calls(void **state)  
{  
    (void)state;  
    ignore_function_calls(mock_test_a_called);  
}  
  
static void test_ordering_does_expect_at_least_one_call(void **state)
```

```
{  
    (void)state;  
    expect_function_call_any(mock_test_a_called);  
  
    mock_test_a_called();  
    mock_test_a_called();  
    mock_test_a_called();  
}  
  
static void test_ordering_does_work_across_different_functions(void **state)  
{  
    (void)state;  
    expect_function_call(mock_test_a_called);  
    expect_function_call(mock_test_b_called);  
    expect_function_call(mock_test_a_called);  
  
    mock_test_a_called();  
    mock_test_b_called();  
    mock_test_a_called();  
}  
  
static void test_ordering_ignores_out_of_order_properly(void **state)  
{  
    (void)state;  
    ignore_function_calls(mock_test_a_called);  
    ignore_function_calls(mock_test_b_called);  
    expect_function_calls(mock_test_c_called, 2);
```

```

mock_test_c_called();
mock_test_b_called();
mock_test_c_called();
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_does_succeed_for_expected)
        ,cmocka_unit_test(test_does_succeed_for_multiple_calls)
        ,cmocka_unit_test(test_ordering_does_ignore_no_calls)
        ,cmocka_unit_test(test_ordering_does_ignore_calls)
        ,cmocka_unit_test(test_ordering_does_expect_at_least_one_call)
        ,cmocka_unit_test(test_ordering_does_work_across_different_functions)
        ,cmocka_unit_test(test_ordering_ignores_out_of_order_properly)
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

#define UNIT_TESTING 1

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

static int setup_fail(void **state) {
    *state = NULL;

    /* We need to fail in setup */
}

```

```
    return -1;
}

static void int_test_ignored(void **state) {
    /* should not be called */
    assert_non_null(*state);
}

static int setup_ok(void **state) {
    int *answer;

    answer = malloc(sizeof(int));
    if (answer == NULL) {
        return -1;
    }
    *answer = 42;

    *state = answer;

    return 0;
}

/* A test case that does check if an int is equal. */
static void int_test_success(void **state) {
    int *answer = *state;

    assert_int_equal(*answer, 42);
}
```

```
static int teardown(void **state) {
    free(*state);

    return 0;
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test_setup_teardown(int_test_ignored, setup_fail, teardown),
        cmocka_unit_test_setup_teardown(int_test_success, setup_ok, teardown),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>

#include <stdlib.h>

int mock_function(void);
void mock_function_call_times(size_t times, int expectedValue);

int mock_function(void)
{
```

```
return (int) mock();  
}  
  
void mock_function_call_times(size_t times, int expectedValue)  
{  
    size_t i;  
    for (i = 0; i < times; ++i)  
    {  
        assert_int_equal(expectedValue, mock_function());  
    }  
}  
  
static void test_will_return_fails_for_no_calls(void **state)  
{  
    (void) state;  
  
    will_return(mock_function, 32);  
}  
  
static void test_will_return_count_fails_for_unreturned_items(void **state)  
{  
    int value;  
    size_t numberOfCalls;  
  
    (void) state;  
  
    value = rand();  
    numberOfCalls = (size_t) ((rand() % 20 + 2);
```

```
    will_return_count(mock_function, value, numberOfCalls);
    mock_function_call_times(numberOfCalls - 1u, value);
}

static void test_will_return_always_fails_for_no_calls(void **state)
{
    int value;

    (void) state;

    value = rand();

    will_return_always(mock_function, value);
}

static int teardown(void **state) {
    free(*state);

    return 0;
}

int main(int argc, char **argv) {
    const struct CMUnitTest alloc_tests[] = {
        cmocka_unit_test_teardown(test_will_return_fails_for_no_calls, teardown)
        ,cmocka_unit_test_teardown(test_will_return_count_fails_for_unreturned_items, teardown)
        ,cmocka_unit_test_teardown(test_will_return_always_fails_for_no_calls, teardown)
    };

    (void)argc;
```

```
(void)argv;

    return cmocka_run_group_tests(alloc_tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/* Use the unit test allocators */
#define UNIT_TESTING 1

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

static int setup(void **state) {
```

```
int *answer = malloc(sizeof(int));

assert_non_null(answer);
*answer = 42;

*state = answer;

return 0;
}

static int teardown(void **state) {
    free(*state);

    return 0;
}

/* A test case that does nothing and succeeds.*/
static void null_test_success(void **state) {
    (void) state;
}

/* A test case that does check if an int is equal.*/
static void int_test_success(void **state) {
    int *answer = *state;

    assert_int_equal(*answer, 42);
}
```

```
int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(null_test_success),
        cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

#include "config.h"

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static void torture_test_malloc(void **state)
{
    char *str;
    size_t str_len;
    size_t len;

    (void)state; /* unused */

    str_len = 12;
```

```
str = (char *)test_malloc(str_len);
assert_non_null(str);

len = sprintf(str, str_len, "test string");
assert_int_equal(len, 11);

len = strlen(str);
assert_int_equal(len, 11);

test_free(str);
}

static void torture_test_realloc(void **state)
{
    char *str;
    char *tmp;
    size_t str_len;
    size_t len;

    (void)state; /* unused */

    str_len = 16;
    str = (char *)test_malloc(str_len);
    assert_non_null(str);

    len = sprintf(str, str_len, "test string 123");
    assert_int_equal(len, 15);

    len = strlen(str);
```

```
assert_int_equal(len, 15);

str_len = 20;
tmp = test_realloc(str, str_len);
assert_non_null(tmp);

str = tmp;
len = strlen(str);
assert_string_equal(tmp, "test string 123");

snprintf(str + len, str_len - len, "4567");
assert_string_equal(tmp, "test string 1234567");

test_free(str);
}
```

```
static void torture_test_realloc_set0(void **state)
{
    char *str;
    size_t str_len;

    (void)state; /* unused */

    str_len = 16;
    str = (char *)test_malloc(str_len);
    assert_non_null(str);

    /* realloc(ptr, 0) is like a free() */
    str = (char *)test_realloc(str, 0);
```

```
    assert_null(str);
}

int main(void) {
    const struct CMUnitTest alloc_tests[] = {
        cmocka_unit_test(torture_test_malloc),
        cmocka_unit_test(torture_test_realloc),
        cmocka_unit_test(torture_test_realloc_set0),
    };

    return cmocka_run_group_tests(alloc_tests, NULL, NULL);
}

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

#include <stdlib.h>

static int setup_only(void **state)
{
    *state = malloc(1);

    return 0;
}

static int teardown_only(void **state)
{
    free(*state);
}
```

```
    return 0;
}

static void malloc_setup_test(void **state)
{
    assert_non_null(*state);
    free(*state);
}

static void malloc_teardown_test(void **state)
{
    *state = malloc(1);
    assert_non_null(*state);
}

static int prestate_setup(void **state)
{
    int *val = (int *)*state, *a;

    a = malloc(sizeof(int));
    *a = *val + 1;
    *state = a;

    return 0;
}

static int prestate_teardown(void **state)
{
```

```
    free(*state);

    return 0;
}

static void prestate_setup_test(void **state)
{
    int *a = (int *)*state;

    assert_non_null(a);
    assert_int_equal(*a, 43);
}

static void prestate_test(void **state)
{
    int *a = (int *)*state;

    assert_non_null(a);
    assert_int_equal(*a, 42);
}

int main(void) {
    int prestate = 42;
    const struct CMUnitTests tests[] = {
        cmocka_unit_test_setup(malloc_setup_test, setup_only),
        cmocka_unit_test_setup(malloc_setup_test, setup_only),
        cmocka_unit_test_teardown(malloc_teardown_test, teardown_only),
        cmocka_unit_test_teardown(malloc_teardown_test, teardown_only),
        cmocka_unit_test_teardown(malloc_teardown_test, teardown_only),
    };
}
```

```
    cmocka_unit_test_teardown(malloc_teardown_test, teardown_only),
    cmocka_unit_test_prestate(prestate_test, &prestate),
    cmocka_unit_test_prestate_setup_teardown(prestate_setup_test, prestate_setup,
prestate_teardown, &prestate),
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2014 Luis Pabon, Jr.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
 * Programming by Contract is a programming methodology
 * which binds the caller and the function called to a
 * contract. The contract is represented using Hoare Triple:
 * {P} C {Q}

```

- * where {P} is the precondition before executing command C,
- * and {Q} is the postcondition.

*

- * See also:

- * http://en.wikipedia.org/wiki/Design_by_contract

- * http://en.wikipedia.org/wiki/Hoare_logic

- * <http://dlang.org/dbc.html>

*/

```
#ifndef CMOCKA_PBC_H_
```

```
#define CMOCKA_PBC_H_
```

```
#if defined(UNIT_TESTING) || defined (DEBUG)
```

```
#include <assert.h>
```

/*

- * Checks caller responsibility against contract

*/

```
#define REQUIRE(cond) assert(cond)
```

/*

- * Checks function responsibility against contract.

*/

```
#define ENSURE(cond) assert(cond)
```

/*

- * While REQUIRE and ENSURE apply to functions, INVARIANT

- * applies to classes/structs. It ensures that instances

- * of the class/struct are consistent. In other words,

```
* that the instance has not been corrupted.  
*/  
  
#define INVARIANT(invariant_fnc) do{ (invariant_fnc) } while (0);  
  
#else  
#define REQUIRE(cond) do { } while (0);  
#define ENSURE(cond) do { } while (0);  
#define INVARIANT(invariant_fnc) do{ } while (0);  
  
#endif /* defined(UNIT_TESTING) || defined (DEBUG) */  
#endif /* CMOCKA_PBC_H_ */  
  
/*  
 * Copyright 2008 Google Inc.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 * http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
*/  
  
#ifndef CMOCKA_H_  
#define CMOCKA_H_
```

```
#ifdef _WIN32
#define __func__ __FUNCTION__

#ifndef inline
#define inline __inline
#endif /* inline */

#ifndef _MSC_VER < 1500
#ifndef __cplusplus
extern "C" {
#endif /* __cplusplus */
int __stdcall IsDebuggerPresent();
#ifndef __cplusplus
} /* extern "C" */
#endif /* __cplusplus */
#endif /* _MSC_VER < 1500 */
#endif /* _MSC_VER */
#endif /* _WIN32 */

/***
 * @defgroup cmocka The CMocka API
 *
 * These headers or their equivalents should be included prior to including
 * this header file.
 * @code
 * #include <stdarg.h>
 */
```

```
* #include <stddef.h>
* #include <setjmp.h>
* @endcode
*
* This allows test applications to use custom definitions of C standard
* library functions and types.
*
* @{
*/
/* If __WORDSIZE is not set, try to figure it out and default to 32 bit. */
#ifndef __WORDSIZE
# if defined(__x86_64__) && !defined(__ILP32__)
# define __WORDSIZE 64
# else
# define __WORDSIZE 32
# endif
#endif

#endif DOXYGEN
/**

* Largest integral type. This type should be large enough to hold any
* pointer or integer supported by the compiler.
*/
typedef uintmax_t LargestIntegralType;
#else /* DOXGEN */
#ifndef LargestIntegralType
# if __WORDSIZE == 64
# define LargestIntegralType unsigned long int

```

```
# else

# define LargestIntegralType unsigned long long int

# endif

#endif /* LargestIntegralType */

#endif /* DOXYGEN */


/* Printf format used to display LargestIntegralType as a hexadecimal. */

#ifndef LargestIntegralTypePrintfFormat

#ifndef _WIN32

# define LargestIntegralTypePrintfFormat "0x%I64x"

#else

# if __WORDSIZE == 64

# define LargestIntegralTypePrintfFormat "%#lx"

#else

# define LargestIntegralTypePrintfFormat "%#llx"

#endif

#endif /* _WIN32 */

#endif /* LargestIntegralTypePrintfFormat */


/* Printf format used to display LargestIntegralType as a decimal. */

#ifndef LargestIntegralTypePrintfFormatDecimal

#ifndef _WIN32

# define LargestIntegralTypePrintfFormatDecimal "%I64u"

#else

# if __WORDSIZE == 64

# define LargestIntegralTypePrintfFormatDecimal "%lu"

#else

# define LargestIntegralTypePrintfFormatDecimal "%llu"

#endif

#endif
```

```
# endif /* _WIN32 */

#endif /* LargestIntegralTypePrintfFormat */

/* Perform an unsigned cast to LargestIntegralType. */

#define cast_to_largest_integral_type(value) \
    ((LargestIntegralType)(value))

/* Smallest integral type capable of holding a pointer. */

#if !defined(_UINTPTR_T) && !defined(_UINTPTR_T_DEFINED)
# if defined(_WIN32)
    /* WIN32 is an ILP32 platform */
    typedef unsigned int uintptr_t;
# elif defined(_WIN64)
    typedef unsigned long int uintptr_t
# else /* _WIN32 */
    /* ILP32 and LP64 platforms */
    # ifdef __WORDSIZE /* glibc */
        # if __WORDSIZE == 64
            typedef unsigned long int uintptr_t;
        # else
            typedef unsigned int uintptr_t;
        # endif /* __WORDSIZE == 64 */
    # else /* __WORDSIZE */
        # if defined(_LP64) || defined(_I32LPx)
            typedef unsigned long int uintptr_t;
        # else
            typedef unsigned int uintptr_t;
        # endif
    # endif

```

```

# endif /* __WORDSIZE */

# endif /* _WIN32 */


#define _UINTPTR_T
#define _UINTPTR_T_DEFINED
#endif /* !defined(_UINTPTR_T) || !defined(_UINTPTR_T_DEFINED) */


/* Perform an unsigned cast to uintptr_t. */
#define cast_to_pointer_integral_type(value) \
((uintptr_t)((size_t)(value)))

/* Perform a cast of a pointer to LargestIntegralType */
#define cast_ptr_to_largest_integral_type(value) \
cast_to_largest_integral_type(cast_to_pointer_integral_type(value))

/* GCC have printf type attribute check. */
#ifndef __GNUC__
#define CMOCKA_PRINTF_ATTRIBUTE(a,b) \
__attribute__((__format__(__printf__, a, b)))
#else
#define CMOCKA_PRINTF_ATTRIBUTE(a,b)
#endif /* __GNUC__ */

#if defined(__GNUC__)
#define CMOCKA_DEPRECATED __attribute__((deprecated))
#elif defined(_MSC_VER)
#define CMOCKA_DEPRECATED __declspec(deprecated)
#else
#define CMOCKA_DEPRECATED

```

```
#endif

#define WILL_RETURN_ALWAYS -1
#define WILL_RETURN_ONCE -2

/***
 * @defgroup cmocka_mock Mock Objects
 * @ingroup cmocka
 *
 * Mock objects mock objects are simulated objects that mimic the behavior of
 * real objects. Instead of calling the real objects, the tested object calls a
 * mock object that merely asserts that the correct methods were called, with
 * the expected parameters, in the correct order.
 *
 * <ul>
 * <li><strong>will_return(function, value)</strong> - The will_return() macro
 * pushes a value onto a stack of mock values. This macro is intended to be
 * used by the unit test itself, while programming the behaviour of the mocked
 * object.</li>
 *
 * <li><strong>mock()</strong> - the mock macro pops a value from a stack of
 * test values. The user of the mock() macro is the mocked object that uses it
 * to learn how it should behave.</li>
 *
 * </ul>
 *
 * Because the will_return() and mock() are intended to be used in pairs, the
 * cmocka library would fail the test if there are more values pushed onto the
 * stack using will_return() than consumed with mock() and vice-versa.
 *
*/
```

```
* The following unit test stub illustrates how would a unit test instruct the
* mock object to return a particular value:
*
* @code
* will_return(chef_cook, "hotdog");
* will_return(chef_cook, 0);
* @endcode
*
* Now the mock object can check if the parameter it received is the parameter
* which is expected by the test driver. This can be done the following way:
*
* @code
* int chef_cook(const char *order, char **dish_out)
* {
*     check_expected(order);
* }
* @endcode
*
* For a complete example please at a look
* <a href="http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef_wrap/waiter_test_wrap.c">her
e</a>.
*
* @{
*/
#endif DOXYGEN
/**

* @brief Retrieve a return value of the current function.
```

```
*  
* @return The value which was stored to return by this function.  
*  
* @see will_return()  
*/  
  
LargestIntegralType mock(void);  
  
#else  
  
#define mock() _mock(__func__, __FILE__, __LINE__)  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Retrieve a typed return value of the current function.  
 *  
 * The value would be casted to type internally to avoid having the  
 * caller to do the cast manually.  
 *  
 * @param[in] #type The expected type of the return value  
 *  
 * @return The value which was stored to return by this function.  
 *  
 * @code  
 * int param;  
 *  
 * param = mock_type(int);  
 * @endcode  
 *  
 * @see will_return()  
 * @see mock()  
*/
```

```
* @see mock_ptr_type()  
*/  
  
#type mock_type(#type);  
  
#else  
  
#define mock_type(type) ((type) mock())  
  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Retrieve a typed return value of the current function.  
 *  
 * The value would be casted to type internally to avoid having the  
 * caller to do the cast manually but also casted to uintptr_t to make  
 * sure the result has a valid size to be used as a pointer.  
 *  
 * @param[in] #type The expected type of the return value  
 *  
 * @return The value which was stored to return by this function.  
 *  
 * @code  
 * char *param;  
 *  
 * param = mock_ptr_type(char *);  
 * @endcode  
 *  
 * @see will_return()  
 * @see mock()  
 * @see mock_type()  
*/
```

```
type mock_ptr_type(#type);

#else

#define mock_ptr_type(type) ((type) (uintptr_t) mock())

#endif


#ifndef DOXYGEN

/**

 * @brief Store a value to be returned by mock() later.

 *

 * @param[in] #function The function which should return the given value.

 *

 * @param[in] value The value to be returned by mock().

 *

 * @code

 * int return_integer(void)

 * {

 *     return (int)mock();

 * }

 *

 * static void test_integer_return(void **state)

 * {

 *     will_return(return_integer, 42);

 *

 *     assert_int_equal(my_function_calling_return_integer(), 42);

 * }

 *

 * @endcode

 *

 * @see mock()


```

```

* @see will_return_count()

*/
void will_return(#function, LargestIntegralType value);

#else

#define will_return(function, value) \
    _will_return(#function, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), 1)

#endif

#ifndef DOXYGEN

/**

* @brief Store a value to be returned by mock() later.

*
* @param[in] #function The function which should return the given value.

*
* @param[in] value The value to be returned by mock().

*
* @param[in] count The parameter indicates the number of times the value should
*                 be returned by mock(). If count is set to -1, the value
*                 will always be returned but must be returned at least once.
*                 If count is set to -2, the value will always be returned
*                 by mock(), but is not required to be returned.
*
* @see mock()

*/
void will_return_count(#function, LargestIntegralType value, int count);

#else

#define will_return_count(function, value, count) \
    _will_return(#function, __FILE__, __LINE__, \

```

```
    cast_to_largest_integral_type(value), count)

#endifif

#ifndef DOXYGEN

/**

 * @brief Store a value that will be always returned by mock().

 *

 * @param[in] #function The function which should return the given value.

 *

 * @param[in] #value The value to be returned by mock().

 *

 * This is equivalent to:

 * @code

 * will_return_count(function, value, -1);

 * @endcode

 *

 * @see will_return_count()

 * @see mock()

 */

void will_return_always(#function, LargestIntegralType value);

#else

#define will_return_always(function, value) \
    will_return_count(function, (value), WILL_RETURN_ALWAYS)

#endifif

#ifndef DOXYGEN

/**

 * @brief Store a value that may be always returned by mock().

 *
```

```
* This stores a value which will always be returned by mock() but is not
* required to be returned by at least one call to mock(). Therefore,
* in contrast to will_return_always() which causes a test failure if it
* is not returned at least once, will_return_maybe() will never cause a test
* to fail if its value is not returned.

*
* @param[in] #function The function which should return the given value.

*
* @param[in] #value The value to be returned by mock().

*
* This is equivalent to:
* @code
* will_return_count(function, value, -2);
* @endcode
*
* @see will_return_count()
* @see mock()
*/
void will_return_maybe(#function, LargestIntegralType value);

#else
#define will_return_maybe(function, value) \
    will_return_count(function, (value), WILL_RETURN_ONCE)

#endif
/** @} */

/**
* @defgroup cmocka_param Checking Parameters
* @ingroup cmocka
*

```

- * Functionality to store expected values for mock function parameters.
- *
- * In addition to storing the return values of mock functions, cmocka provides
- * functionality to store expected values for mock function parameters using
- * the expect_() functions provided. A mock function parameter can then be
- * validated using the check_expected() macro.
- *
- * Successive calls to expect_() macros for a parameter queues values to check
- * the specified parameter. check_expected() checks a function parameter
- * against the next value queued using expect_(), if the parameter check fails
- * a test failure is signalled. In addition if check_expected() is called and
- * no more parameter values are queued a test failure occurs.
- *
- * The following test stub illustrates how to do this. First is the the function
- * we call in the test driver:
- *
- * @code
- * static void test_driver(void **state)
- * {
- * expect_string(chef_cook, order, "hotdog");
- * }
- * @endcode
- *
- * Now the chef_cook function can check if the parameter we got passed is the
- * parameter which is expected by the test driver. This can be done the
- * following way:
- *
- * @code
- * int chef_cook(const char *order, char **dish_out)

```
* {
*   check_expected(order);
*
* @endcode
*
* For a complete example please at a look at
*
* <a href="http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef_wrap/waiter_test_wrap.c">here</a>
*
* @{
*
/* Add a custom parameter checking function. If the event parameter is NULL
* the event structure is allocated internally by this function. If event
* parameter is provided it must be allocated on the heap and doesn't need to
* be deallocated by the caller.
*/
#endif DOXYGEN
/** @brief Add a custom parameter checking function.
*
* If the event parameter is NULL the event structure is allocated internally
* by this function. If the parameter is provided it must be allocated on the
* heap and doesn't need to be deallocated by the caller.
*
* @param[in] #function The function to add a custom parameter checking
*                   function for.

```

```

*
* @param[in] #parameter The parameters passed to the function.
*
* @param[in] #check_function The check function to call.
*
* @param[in] check_data      The data to pass to the check function.
*/
void expect_check(#function, #parameter, #check_function, const void *check_data);

#else
#define expect_check(function, parameter, check_function, check_data) \
    _expect_check(#function, #parameter, __FILE__, __LINE__, check_function, \
        cast_to_largest_integral_type(check_data), NULL, 1)
#endif

#ifndef DOXYGEN
/** 
* @brief Add an event to check if the parameter value is part of the provided
*       array.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @see check_expected().
*/

```

```

void expect_in_set(#function, #parameter, LargestIntegralType value_array[]);

#else

#define expect_in_set(function, parameter, value_array) \
    expect_in_set_count(function, parameter, value_array, 1)

#endif


#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter value is part of the provided
 *        array.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] value_array[] The array to check for the value.
 *
 * @param[in] count The count parameter returns the number of times the value
 *                 should be returned by check_expected(). If count is set
 *                 to -1 the value will always be returned.
 *
 * @see check_expected().
 */

void expect_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);

#else

#define expect_in_set_count(function, parameter, value_array, count) \
    _expect_in_set(#function, #parameter, __FILE__, __LINE__, value_array, \

```

```
    sizeof(value_array) / sizeof((value_array)[0]), count)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter value is not part of the
 *        provided array.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *

 * @param[in] #parameter The name of the parameter passed to the function.

 *

 * @param[in] value_array[] The array to check for the value.

 *

 * @see check_expected().

 */

void expect_not_in_set(#function, #parameter, LargestIntegralType value_array[]);

#else

#define expect_not_in_set(function, parameter, value_array) \
    expect_not_in_set_count(function, parameter, value_array, 1)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter value is not part of the
 *        provided array.

 *
```

```

* The event is triggered by calling check_expected() in the mocked function.

*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);

#define expect_not_in_set(function, parameter, value_array, count) \
    _expect_not_in_set( \
        #function, #parameter, __FILE__, __LINE__, value_array, \
        sizeof(value_array) / sizeof((value_array)[0]), count)

#endif

#ifndef DOXYGEN
/**

* @brief Add an event to check a parameter is inside a numerical range.
* The check would succeed if minimum <= value <= maximum.
*
* The event is triggered by calling check_expected() in the mocked function.

```

```

*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @see check_expected().
*/
void expect_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum);

#else
#define expect_in_range(function, parameter, minimum, maximum) \
    expect_in_range_count(function, parameter, minimum, maximum, 1)
#endif

#ifndef DOXYGEN
/** 
* @brief Add an event to repeatedly check a parameter is inside a
* numerical range. The check would succeed if minimum <= value <= maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*

```

```

* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_in_range_count(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum, size_t count);

#else

#define expect_in_range_count(function, parameter, minimum, maximum, count) \
_expect_in_range(#function, #parameter, __FILE__, __LINE__, minimum, \
maximum, count)

#endif

#ifndef DOXYGEN

/**

* @brief Add an event to check a parameter is outside a numerical range.
* The check would succeed if minimum > value > maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*

```

```

* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @see check_expected().
*/
void expect_not_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum);

#else
#define expect_not_in_range(function, parameter, minimum, maximum) \
expect_not_in_range_count(function, parameter, minimum, maximum, 1)
#endif

#ifndef DOXYGEN
/**

* @brief Add an event to repeatedly check a parameter is outside a
* numerical range. The check would succeed if minimum > value > maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @param[in] count The count parameter returns the number of times the value

```

```

*      should be returned by check_expected(). If count is set
*      to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_in_range_count(#function, #parameter, LargestIntegralType minimum,
LargestIntegralType maximum, size_t count);

#else

#define expect_not_in_range_count(function, parameter, minimum, maximum, \
                                count) \
    _expect_not_in_range(#function, #parameter, __FILE__, __LINE__, \
                        minimum, maximum, count)

#endif

#ifndef DOXYGEN

/**

* @brief Add an event to check if a parameter is the given value.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value The value to check.
*
* @see check_expected().
*/
void expect_value(#function, #parameter, LargestIntegralType value);

```

```

#else

#define expect_value(function, parameter, value) \
    expect_value_count(function, parameter, value, 1)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to repeatedly check if a parameter is the given value.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] value The value to check.
 *
 * @param[in] count The count parameter returns the number of times the value
 *                 should be returned by check_expected(). If count is set
 *                 to -1 the value will always be returned.
 *
 * @see check_expected().
 */

void expect_value_count(#function, #parameter, LargestIntegralType value, size_t count);

#else

#define expect_value_count(function, parameter, value, count) \
    _expect_value(#function, #parameter, __FILE__, __LINE__, \
                  cast_to_largest_integral_type(value), count)

#endif

```

```
#ifdef DOXYGEN

/**

 * @brief Add an event to check if a parameter isn't the given value.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *

 * @param[in] #parameter The name of the parameter passed to the function.

 *

 * @param[in] value The value to check.

 *

 * @see check_expected().

 */

void expect_not_value(#function, #parameter, LargestIntegralType value);

#else

#define expect_not_value(function, parameter, value) \
    expect_not_value_count(function, parameter, value, 1)

#endif
```

```
#ifdef DOXYGEN

/**

 * @brief Add an event to repeatedly check if a parameter isn't the given value.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *
```

```

* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value The value to check.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_value_count(#function, #parameter, LargestIntegralType value, size_t count);
#else
#define expect_not_value_count(function, parameter, value, count) \
    _expect_not_value(#function, #parameter, __FILE__, __LINE__, \
                      cast_to_largest_integral_type(value), count)
#endif

#ifndef DOXYGEN
/***
* @brief Add an event to check if the parameter value is equal to the
*        provided string.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.

```

```

*
* @see check_expected().
*/
void expect_string(#function, #parameter, const char *string);
#else
#define expect_string(function, parameter, string) \
    expect_string_count(function, parameter, string, 1)
#endif

#ifndef DOXYGEN
/***
* @brief Add an event to check if the parameter value is equal to the
*        provided string.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.
*
* @param[in] count The count parameter returns the number of times the value
*        should be returned by check_expected(). If count is set
*        to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_string_count(#function, #parameter, const char *string, size_t count);

```

```

#else

#define expect_string_count(function, parameter, string, count) \
    _expect_string(#function, #parameter, __FILE__, __LINE__, \
        (const char*)(string), count)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter value isn't equal to the
 * provided string.

 *
 * The event is triggered by calling check_expected() in the mocked function.

 *
 * @param[in] #function The function to add the check for.

 *
 * @param[in] #parameter The name of the parameter passed to the function.

 *
 * @param[in] string The string value to compare.

 *
 * @see check_expected().
 */

void expect_not_string(#function, #parameter, const char *string);

#else

#define expect_not_string(function, parameter, string) \
    expect_string_count(function, parameter, string, 1)

#endif

#ifndef DOXYGEN

/**
```

```
* @brief Add an event to check if the parameter value isn't equal to the
*       provided string.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.
*
* @param[in] count The count parameter returns the number of times the value
*       should be returned by check_expected(). If count is set
*       to -1 the value will always be returned.
*
* @see check_expected().
*/

```

```
void expect_not_string_count(#function, #parameter, const char *string, size_t count);
#else
#define expect_not_string_count(function, parameter, string, count) \
    _expect_not_string(#function, #parameter, __FILE__, __LINE__, \
        (const char*)(string), count)
#endif
```

```
#ifdef DOXYGEN
/***
* @brief Add an event to check if the parameter does match an area of memory.
*
* The event is triggered by calling check_expected() in the mocked function.
```

```
*  
* @param[in] #function The function to add the check for.  
*  
* @param[in] #parameter The name of the parameter passed to the function.  
*  
* @param[in] memory The memory to compare.  
*  
* @param[in] size The size of the memory to compare.  
*  
* @see check_expected().  
*/  
  
void expect_memory(#function, #parameter, void *memory, size_t size);  
  
#else  
  
#define expect_memory(function, parameter, memory, size) \  
    expect_memory_count(function, parameter, memory, size, 1)  
  
#endif  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Add an event to repeatedly check if the parameter does match an area  
 *        of memory.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] memory The memory to compare.
```

```

*
* @param[in] size The size of the memory to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_memory_count(#function, #parameter, void *memory, size_t size, size_t count);
#else
#define expect_memory_count(function, parameter, memory, size, count) \
    _expect_memory(#function, #parameter, __FILE__, __LINE__, \
                  (const void*)(memory), size, count)
#endif

#ifndef DOXYGEN
/** 
* @brief Add an event to check if the parameter doesn't match an area of
*        memory.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] memory The memory to compare.
*

```

```
* @param[in] size The size of the memory to compare.  
*  
* @see check_expected().  
*/  
  
void expect_not_memory(#function, #parameter, void *memory, size_t size);  
  
#else  
  
#define expect_not_memory(function, parameter, memory, size) \  
    expect_not_memory_count(function, parameter, memory, size, 1)  
  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Add an event to repeatedly check if the parameter doesn't match an  
 *        area of memory.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] memory The memory to compare.  
 *  
 * @param[in] size The size of the memory to compare.  
 *  
 * @param[in] count The count parameter returns the number of times the value  
 *        should be returned by check_expected(). If count is set  
 *        to -1 the value will always be returned.  
 */
```

```

* @see check_expected().

*/
void expect_not_memory_count(#function, #parameter, void *memory, size_t size, size_t count);
#else
#define expect_not_memory_count(function, parameter, memory, size, count) \
_expect_not_memory(#function, #parameter, __FILE__, __LINE__, \
(const void*)(memory), size, count)

#endif

#endif DOXYGEN

/***
* @brief Add an event to check if a parameter (of any value) has been passed.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @see check_expected().
*/
void expect_any(#function, #parameter);
#else
#define expect_any(function, parameter) \
expect_any_count(function, parameter, 1)
#endif

#endif DOXYGEN

```

```
/**  
 * @brief Add an event to repeatedly check if a parameter (of any value) has  
 * been passed.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] count The count parameter returns the number of times the value  
 * should be returned by check_expected(). If count is set  
 * to -1 the value will always be returned.  
 *  
 * @see check_expected().
```

```
*/  
  
void expect_any_count(#function, #parameter, size_t count);  
  
#else  
  
#define expect_any_count(function, parameter, count) \  
    _expect_any(#function, #parameter, __FILE__, __LINE__, count)  
  
#endif
```

```
#ifdef DOXYGEN  
  
/**  
 * @brief Determine whether a function parameter is correct.  
 *  
 * This ensures the next value queued by one of the expect_*() macros matches  
 * the specified variable.  
 *
```

```

* This function needs to be called in the mock object.

*
* @param[in] #parameter The parameter to check.
*/
void check_expected(#parameter);

#else
#define check_expected(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(parameter))

#endif

#ifndef DOXYGEN
/** 
* @brief Determine whether a function parameter is correct.
*
* This ensures the next value queued by one of the expect_*() macros matches
* the specified variable.
*
* This function needs to be called in the mock object.
*
* @param[in] #parameter The pointer to check.
*/
void check_expected_ptr(#parameter);

#else
#define check_expected_ptr(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
        cast_ptr_to_largest_integral_type(parameter))

#endif

```

```
/** @} */

/**
 * @defgroup cmocka_asserts Assert Macros
 * @ingroup cmocka
 *
 * This is a set of useful assert macros like the standard C library's
 * assert(3) macro.
 *
 * On an assertion failure a cmocka assert macro will write the failure to the
 * standard error stream and signal a test failure. Due to limitations of the C
 * language the general C standard library assert() and cmocka's assert_true()
 * and assert_false() macros can only display the expression that caused the
 * assert failure. cmocka's type specific assert macros, assert_{type}_equal()
 * and assert_{type}_not_equal(), display the data that caused the assertion
 * failure which increases data visibility aiding debugging of failing test
 * cases.
 *
 * @{
 */

```

```
#ifdef DOXYGEN

/**
 * @brief Assert that the given expression is true.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if expression is false (i.e., compares equal to
 * zero).
 *
```



```
#ifdef DOXYGEN

/**

 * @brief Assert that the return_code is greater than or equal to 0.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the return code is smaller than 0. If the function

 * you check sets an errno if it fails you can pass it to the function and

 * it will be printed as part of the error message.

 *

 * @param[in] rc      The return code to evaluate.

 *

 * @param[in] error   Pass errno here or 0.

 */

void assert_return_code(int rc, int error);

#else

#define assert_return_code(rc, error) \
    _assert_return_code(cast_to_largest_integral_type(rc), \
                        sizeof(rc), \
                        cast_to_largest_integral_type(error), \
                        #rc, __FILE__, __LINE__)

#endif
```

```
#ifdef DOXYGEN

/**

 * @brief Assert that the given pointer is non-NULL.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the pointer is non-NULL.

 *
```

```

* @param[in] pointer The pointer to evaluate.
*
* @see assert_null()
*/
void assert_non_null(void *pointer);

#else

#define assert_non_null(c) _assert_true(cast_ptr_to_largest_integral_type(c), #c, \
                                         __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Assert that the given pointer is NULL.

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the pointer is non-NULL.

*
* @param[in] pointer The pointer to evaluate.

*
* @see assert_non_null()
*/
void assert_null(void *pointer);

#else

#define assert_null(c) _assert_true(!(cast_ptr_to_largest_integral_type(c)), #c, \
                                         __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**
```

```
* @brief Assert that the two given pointers are equal.  
*  
* The function prints an error message and terminates the test by calling  
* fail() if the pointers are not equal.  
*  
* @param[in] a      The first pointer to compare.  
*  
* @param[in] b      The pointer to compare against the first one.  
*/  
  
void assert_ptr_equal(void *a, void *b);  
  
#else  
  
#define assert_ptr_equal(a, b) \  
    _assert_int_equal(cast_ptr_to_largest_integral_type(a), \  
                      cast_ptr_to_largest_integral_type(b), \  
                      __FILE__, __LINE__)  
  
#endif  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Assert that the two given pointers are not equal.  
 *  
 * The function prints an error message and terminates the test by calling  
 * fail() if the pointers are equal.  
 *  
 * @param[in] a      The first pointer to compare.  
 *  
 * @param[in] b      The pointer to compare against the first one.  
*/  
  
void assert_ptr_not_equal(void *a, void *b);
```

```
#else

#define assert_ptr_not_equal(a, b) \
    _assert_int_not_equal(cast_ptr_to_largest_integral_type(a), \
        cast_ptr_to_largest_integral_type(b), \
        __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the two given integers are equal.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the integers are not equal.

 *

 * @param[in] a The first integer to compare.

 *

 * @param[in] b The integer to compare against the first one.

 */

void assert_int_equal(int a, int b);

#else

#define assert_int_equal(a, b) \
    _assert_int_equal(cast_to_largest_integral_type(a), \
        cast_to_largest_integral_type(b), \
        __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the two given integers are not equal.


```

```

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the integers are equal.
*
* @param[in] a The first integer to compare.
*
* @param[in] b The integer to compare against the first one.
*
* @see assert_int_equal()
*/
void assert_int_not_equal(int a, int b);

#else
#define assert_int_not_equal(a, b) \
    _assert_int_not_equal(cast_to_largest_integral_type(a), \
                          cast_to_largest_integral_type(b), \
                          __FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/** 
* @brief Assert that the two given strings are equal.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the strings are not equal.
*
* @param[in] a The string to check.
*
* @param[in] b The other string to compare.
*/

```

```
void assert_string_equal(const char *a, const char *b);

#else

#define assert_string_equal(a, b) \
    _assert_string_equal((const char*)(a), (const char*)(b), __FILE__, \
                        __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the two given strings are not equal.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the strings are equal.

 *

 * @param[in] a The string to check.

 *

 * @param[in] b The other string to compare.

 */

void assert_string_not_equal(const char *a, const char *b);

#else

#define assert_string_not_equal(a, b) \
    _assert_string_not_equal((const char*)(a), (const char*)(b), __FILE__, \
                           __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the two given areas of memory are equal, otherwise fail.

 *
```

```
* The function prints an error message to standard error and terminates the
* test by calling fail() if the memory is not equal.
*
* @param[in] a The first memory area to compare
*           (interpreted as unsigned char).
*
* @param[in] b The second memory area to compare
*           (interpreted as unsigned char).
*
* @param[in] size The first n bytes of the memory areas to compare.
*/

```

```
void assert_memory_equal(const void *a, const void *b, size_t size);
#else
#define assert_memory_equal(a, b, size) \
    _assert_memory_equal((const void*)(a), (const void*)(b), size, __FILE__, \
                         __LINE__)
#endif
```

```
#ifdef DOXYGEN
/**
 * @brief Assert that the two given areas of memory are not equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the memory is equal.
 *
 * @param[in] a The first memory area to compare
 *           (interpreted as unsigned char).
 *
 * @param[in] b The second memory area to compare
```

```

*      (interpreted as unsigned char).

*
* @param[in] size The first n bytes of the memory areas to compare.
*/
void assert_memory_not_equal(const void *a, const void *b, size_t size);

#else
#define assert_memory_not_equal(a, b, size) \
    _assert_memory_not_equal((const void*)(a), (const void*)(b), size, \
                           __FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/**

* @brief Assert that the specified value is not smaller than the minimum
* and and not greater than the maximum.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is not in range.
*
* @param[in] value The value to check.
*
* @param[in] minimum The minimum value allowed.
*
* @param[in] maximum The maximum value allowed.
*/
void assert_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType
maximum);

#else
#define assert_in_range(value, minimum, maximum) \

```

```

_assert_in_range( \
    cast_to_largest_integral_type(value), \
    cast_to_largest_integral_type(minimum), \
    cast_to_largest_integral_type(maximum), __FILE__, __LINE__)

#endif

#ifndef DOXYGEN
/**

* @brief Assert that the specified value is smaller than the minimum or
* greater than the maximum.

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is in range.

*
* @param[in] value The value to check.

*
* @param[in] minimum The minimum value to compare.

*
* @param[in] maximum The maximum value to compare.

*/
void assert_not_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType
maximum);

#else
#define assert_not_in_range(value, minimum, maximum) \
_assert_not_in_range( \
    cast_to_largest_integral_type(value), \
    cast_to_largest_integral_type(minimum), \
    cast_to_largest_integral_type(maximum), __FILE__, __LINE__)

#endif

```

```
#ifdef DOXYGEN

/** 
 * @brief Assert that the specified value is within a set.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if value is not within a set.
 *
 * @param[in] value The value to look up
 *
 * @param[in] values[] The array to check for the value.
 *
 * @param[in] count The size of the values array.
 */

void assert_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);

#else

#define assert_in_set(value, values, number_of_values) \
    _assert_in_set(value, values, number_of_values, __FILE__, __LINE__)

#endif

#endif DOXYGEN

/** 
 * @brief Assert that the specified value is not within a set.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if value is within a set.
 *
 * @param[in] value The value to look up
 *
```

```
* @param[in] values[] The array to check for the value.  
*  
* @param[in] count The size of the values array.  
*/  
  
void assert_not_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);  
  
#else  
  
#define assert_not_in_set(value, values, number_of_values) \  
    _assert_not_in_set(value, values, number_of_values, __FILE__, __LINE__)  
  
#endif  
  
/** @} */
```

```
/**  
 * @defgroup cmocka_call_order Call Ordering  
 * @ingroup cmocka  
 *  
 * It is often beneficial to make sure that functions are called in an  
 * order. This is independent of mock returns and parameter checking as both  
 * of the aforementioned do not check the order in which they are called from  
 * different functions.  
 *  
 * <ul>  
 * <li><strong>expect_function_call(function)</strong> - The  
 * expect_function_call() macro pushes an expectation onto the stack of  
 * expected calls.</li>  
 *  
 * <li><strong>function_called()</strong> - pops a value from the stack of  
 * expected calls. function_called() is invoked within the mock object  
 * that uses it.  
 */
```

- *
- *
- * expect_function_call() and function_called() are intended to be used in
- * pairs. Cmocka will fail a test if there are more or less expected calls
- * created (e.g. expect_function_call()) than consumed with function_called().
- * There are provisions such as ignore_function_calls() which allow this
- * restriction to be circumvented in tests where mock calls for the code under
- * test are not the focus of the test.
- *
- * The following example illustrates how a unit test instructs cmocka
- * to expect a function_called() from a particular mock,
- * chef_sing():
- *
- * @code
- * void chef_sing(void);
- *
- * void code_under_test()
- * {
- * chef_sing();
- * }
- *
- * void some_test(void **state)
- * {
- * expect_function_call(chef_sing);
- * code_under_test();
- * }
- * @endcode
- *

* The implementation of the mock then must check whether it was meant to

```

* be called by invoking <strong>function_called()</strong>:
*
* @code
* void chef_sing()
* {
*   function_called();
* }
* @endcode
*
* @{
*/
#endif DOXYGEN

/**

* @brief Check that current mocked function is being called in the expected
*       order
*
* @see expect_function_call()
*/
void function_called(void);

#ifndef
#define function_called() _function_called(__func__, __FILE__, __LINE__)
#endif

#endif DOXYGEN

/**

* @brief Store expected call(s) to a mock to be checked by function_called()
*       later.
*

```

```

* @param[in] #function The function which should should be called
*
* @param[in] times number of times this mock must be called
*
* @see function_called()
*/
void expect_function_calls(#function, const int times);

#else
#define expect_function_calls(function, times) \
    _expect_function_call(function, __FILE__, __LINE__, times)
#endif

#ifndef DOXYGEN
/**

* @brief Store expected single call to a mock to be checked by
*       function_called() later.
*
* @param[in] #function The function which should should be called
*
* @see function_called()
*/
void expect_function_call(#function);
#else
#define expect_function_call(function) \
    _expect_function_call(function, __FILE__, __LINE__, 1)
#endif

#ifndef DOXYGEN
/**
```

```

* @brief Expects function_called() from given mock at least once
*
* @param[in] #function The function which should should be called
*
* @see function_called()
*/
void expect_function_call_any(#function);

#else
#define expect_function_call_any(function) \
    _expect_function_call(function, __FILE__, __LINE__, -1)
#endif

#ifndef DOXYGEN
/** 
* @brief Ignores function_called() invocations from given mock function.
*
* @param[in] #function The function which should should be called
*
* @see function_called()
*/
void ignore_function_calls(#function);

#else
#define ignore_function_calls(function) \
    _expect_function_call(function, __FILE__, __LINE__, -2)
#endif

/** @} */
/**
```

```
* @defgroup cmocka_exec Running Tests
* @ingroup cmocka
*
* This is the way tests are executed with CMocka.
*
* The following example illustrates this macro's use with the unit_test macro.
*
* @code
* void Test0(void **state);
* void Test1(void **state);
*
* int main(void)
{
*   const struct CMUnitTest tests[] = {
*     cmocka_unit_test(Test0),
*     cmocka_unit_test(Test1),
*   };
*
*   return cmocka_run_group_tests(tests, NULL, NULL);
}
* @endcode
*
* @{
*/
#endif DOXYGEN
/**
* @brief Forces the test to fail immediately and quit.
*/

```

```
void fail(void);

#ifndef DOXYGEN
#define fail() _fail(__FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/** @brief Forces the test to not be executed, but marked as skipped
 */
void skip(void);

#ifndef DOXYGEN
#define skip() _skip(__FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/** @brief Forces the test to fail immediately and quit, printing the reason.
 *
 * @code
 * fail_msg("This is some error message for test");
 * @endcode
 *
 * or
 *
 * @code
 * char *error_msg = "This is some error message for test";
 * fail_msg("%s", error_msg);
 * @endcode
 */

```

```
void fail_msg(const char *msg, ...);

#else

#define fail_msg(msg, ...) do { \
    print_error("ERROR: " msg "\n", ##__VA_ARGS__); \
    fail(); \
} while (0)

#endif

#ifndef DOXYGEN

/***
 * @brief Generic method to run a single test.
 *
 * @deprecated This function was deprecated in favor of cmocka_run_group_tests
 *
 * @param[in] #function The function to test.
 *
 * @return 0 on success, 1 if an error occurred.
 *
 * @code
 * // A test case that does nothing and succeeds.
 *
 * void null_test_success(void **state) {
 *
 * }
 *
 * int main(void) {
 *
 *     return run_test(null_test_success);
 *
 * }
 * @endcode
 */
int run_test(#function);
```

```
#else

#define run_test(f) _run_test(#f, f, NULL, UNIT_TEST_FUNCTION_TYPE_TEST, NULL)

#endif


static inline void _unit_test_dummy(void **state) {

    (void)state;

}

/** Initializes a UnitTest structure.

 *
 * @deprecated This function was deprecated in favor of cmocka_unit_test
 */
#define unit_test(f) { #f, f, UNIT_TEST_FUNCTION_TYPE_TEST }

#define _unit_test_setup(test, setup) \
{ #test " " #setup, setup, UNIT_TEST_FUNCTION_TYPE_SETUP }

/** Initializes a UnitTest structure with a setup function.

 *
 * @deprecated This function was deprecated in favor of cmocka_unit_test_setup
 */
#define unit_test_setup(test, setup) \
    _unit_test_setup(test, setup), \
    unit_test(test), \
    _unit_test_teardown(test, _unit_test_dummy)

#define _unit_test_teardown(test, teardown) \
{ #test " " #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_TEARDOWN }
```

```
/** Initializes a UnitTest structure with a teardown function.  
 *  
 * @deprecated This function was deprecated in favor of cmocka_unit_test_teardown  
 */  
  
#define unit_test_teardown(test, teardown) \  
    _unit_test_setup(test, _unit_test_dummy), \  
    unit_test(test), \  
    _unit_test_teardown(test, teardown)  
  
/** Initializes a UnitTest structure for a group setup function.  
 *  
 * @deprecated This function was deprecated in favor of cmocka_run_group_tests  
 */  
  
#define group_test_setup(setup) \  
{ "group_" #setup, setup, UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP }  
  
/** Initializes a UnitTest structure for a group teardown function.  
 *  
 * @deprecated This function was deprecated in favor of cmocka_run_group_tests  
 */  
  
#define group_test_teardown(teardown) \  
{ "group_" #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN }  
  
/**  
 * Initialize an array of UnitTest structures with a setup function for a test  
 * and a teardown function. Either setup or teardown can be NULL.  
 *  
 * @deprecated This function was deprecated in favor of  
 * cmocka_unit_test_setup_teardown
```

```

*/
#define unit_test_setup_teardown(test, setup, teardown) \
    _unit_test_setup(test, setup), \
    unit_test(test), \
    _unit_test_teardown(test, teardown)

/** Initializes a CMUnitTest structure. */
#define cmocka_unit_test(f) { #f, f, NULL, NULL, NULL }

/** Initializes a CMUnitTest structure with a setup function. */
#define cmocka_unit_test_setup(f, setup) { #f, f, setup, NULL, NULL }

/** Initializes a CMUnitTest structure with a teardown function. */
#define cmocka_unit_test_teardown(f, teardown) { #f, f, NULL, teardown, NULL }

/**
 * Initialize an array of CMUnitTest structures with a setup function for a test
 * and a teardown function. Either setup or teardown can be NULL.
 */
#define cmocka_unit_test_setup_teardown(f, setup, teardown) { #f, f, setup, teardown, NULL }

/**
 * Initialize a CMUnitTest structure with given initial state. It will be passed
 * to test function as an argument later. It can be used when test state does
 * not need special initialization or was initialized already.
 * @note If the group setup function initialized the state already, it won't be
 * overridden by the initial state defined here.
 */

```

```

#define cmocka_unit_test_prestate(f, state) { #f, f, NULL, NULL, state }

/**
* Initialize a CMUnitTest structure with given initial state, setup and
* teardown function. Any of these values can be NULL. Initial state is passed
* later to setup function, or directly to test if none was given.
* @note If the group setup function initialized the state already, it won't be
* overridden by the initial state defined here.
*/

#define cmocka_unit_test_prestate_setup_teardown(f, setup, teardown, state) { #f, f, setup, teardown,
state }

#define run_tests(tests) _run_tests(tests, sizeof(tests) / sizeof(tests)[0])
#define run_group_tests(tests) _run_group_tests(tests, sizeof(tests) / sizeof(tests)[0])

#ifdef DOXYGEN

/**
* @brief Run tests specified by an array of CMUnitTest structures.
*
* @param[in] group_tests[] The array of unit tests to execute.
*
* @param[in] group_setup The setup function which should be called before
* all unit tests are executed.
*
* @param[in] group_teardown The teardown function to be called after all
* tests have finished.
*
* @return 0 on success, or the number of failed tests.
*

```

```
* @code

* static int setup(void **state) {
*     int *answer = malloc(sizeof(int));
*     if (*answer == NULL) {
*         return -1;
*     }
*     *answer = 42;
*
*     *state = answer;
*
*     return 0;
* }

*
* static int teardown(void **state) {
*     free(*state);
*
*     return 0;
* }

*
* static void null_test_success(void **state) {
*     (void) state;
* }

*
* static void int_test_success(void **state) {
*     int *answer = *state;
*     assert_int_equal(*answer, 42);
* }

*
* int main(void) {
```

```

*   const struct CMUnitTest tests[] = {
*       cmocka_unit_test(null_test_success),
*       cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),
*   };
*
*   return cmocka_run_group_tests(tests, NULL, NULL);
* }
* @endcode
*
* @see cmocka_unit_test
* @see cmocka_unit_test_setup
* @see cmocka_unit_test_teardown
* @see cmocka_unit_test_setup_teardown
*/
int cmocka_run_group_tests(const struct CMUnitTest group_tests[],
                           CMFixtureFunction group_setup,
                           CMFixtureFunction group_teardown);

#else
#define cmocka_run_group_tests(group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(#group_tests, group_tests, sizeof(group_tests) / sizeof(group_tests)[0], \
                           group_setup, group_teardown)

#endif

#ifndef DOXYGEN
/**

* @brief Run tests specified by an array of CMUnitTest structures and specify
*        a name.

*
* @param[in] group_name    The name of the group test.

```

```
*  
* @param[in] group_tests[] The array of unit tests to execute.  
*  
* @param[in] group_setup The setup function which should be called before  
*           all unit tests are executed.  
*  
* @param[in] group_teardown The teardown function to be called after all  
*           tests have finished.  
*  
* @return 0 on success, or the number of failed tests.  
*  
* @code  
* static int setup(void **state) {  
*     int *answer = malloc(sizeof(int));  
*     if (*answer == NULL) {  
*         return -1;  
*     }  
*     *answer = 42;  
*  
*     *state = answer;  
*  
*     return 0;  
* }  
*  
* static int teardown(void **state) {  
*     free(*state);  
*  
*     return 0;  
* }
```



```
#else

#define cmocka_run_group_tests_name(group_name, group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(group_name, group_tests, sizeof(group_tests) / sizeof(group_tests)[0], \
group_setup, group_teardown)

#endif

/** @} */


* @defgroup cmocka_alloc Dynamic Memory Allocation
* @ingroup cmocka
*
* Memory leaks, buffer overflows and underflows can be checked using cmocka.
*
* To test for memory leaks, buffer overflows and underflows a module being
* tested by cmocka should replace calls to malloc(), calloc() and free() to
* test_malloc(), test_calloc() and test_free() respectively. Each time a block
* is deallocated using test_free() it is checked for corruption, if a corrupt
* block is found a test failure is signalled. All blocks allocated using the
* test_*() allocation functions are tracked by the cmocka library. When a test
* completes if any allocated blocks (memory leaks) remain they are reported
* and a test failure is signalled.
*
* For simplicity cmocka currently executes all tests in one process. Therefore
* all test cases in a test application share a single address space which
* means memory corruption from a single test case could potentially cause the
* test application to exit prematurely.
*
* @{

```

```
*/
```

```
#ifdef DOXYGEN
```

```
/**
```

```
* @brief Test function overriding malloc.
```

```
*
```

```
* @param[in] size The bytes which should be allocated.
```

```
*
```

```
* @return A pointer to the allocated memory or NULL on error.
```

```
*
```

```
* @code
```

```
* #ifdef UNIT_TESTING
```

```
* extern void* _test_malloc(const size_t size, const char* file, const int line);
```

```
*
```

```
* #define malloc(size) _test_malloc(size, __FILE__, __LINE__)
```

```
* #endif
```

```
*
```

```
* void leak_memory() {
```

```
*     int * const temporary = (int*)malloc(sizeof(int));
```

```
*     *temporary = 0;
```

```
* }
```

```
* @endcode
```

```
*
```

```
* @see malloc(3)
```

```
*/
```

```
void *test_malloc(size_t size);
```

```
#else
```

```
#define test_malloc(size) _test_malloc(size, __FILE__, __LINE__)
```

```
#endif
```

```
#ifdef DOXYGEN

/** 
 * @brief Test function overriding calloc.
 *
 * The memory is set to zero.
 *
 * @param[in] nmemb The number of elements for an array to be allocated.
 *
 * @param[in] size The size in bytes of each array element to allocate.
 *
 * @return A pointer to the allocated memory, NULL on error.
 *
 * @see calloc(3)
 */

void *test_calloc(size_t nmemb, size_t size);

#else

#define test_calloc(num, size) _test_calloc(num, size, __FILE__, __LINE__)

#endif
```

```
#ifdef DOXYGEN

/** 
 * @brief Test function overriding realloc which detects buffer overruns
 *        and memoery leaks.
 *
 * @param[in] ptr The memory block which should be changed.
 *
 * @param[in] size The bytes which should be allocated.
 */


```

```
* @return      The newly allocated memory block, NULL on error.  
*/  
  
void *test_realloc(void *ptr, size_t size);  
  
#else  
  
#define test_realloc(ptr, size) _test_realloc(ptr, size, __FILE__, __LINE__)  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Test function overriding free(3).  
 *  
 * @param[in] ptr The pointer to the memory space to free.  
 *  
 * @see free(3).  
 */  
  
void test_free(void *ptr);  
  
#else  
  
#define test_free(ptr) _test_free(ptr, __FILE__, __LINE__)  
#endif  
  
  
/* Redirect malloc, calloc and free to the unit test allocators. */  
  
#ifdef UNIT_TESTING  
  
#define malloc test_malloc  
  
#define realloc test_realloc  
  
#define calloc test_calloc  
  
#define free test_free  
  
#endif /* UNIT_TESTING */  
  
  
/** @} */
```

```
/**  
 * @defgroup cmocka_mock_assert Standard Assertions  
 * @ingroup cmocka  
 *  
 * How to handle assert(3) of the standard C library.  
 *  
 * Runtime assert macros like the standard C library's assert() should be  
 * redefined in modules being tested to use cmocka's mock_assert() function.  
 * Normally mock_assert() signals a test failure. If a function is called using  
 * the expect_assert_failure() macro, any calls to mock_assert() within the  
 * function will result in the execution of the test. If no calls to  
 * mock_assert() occur during the function called via expect_assert_failure() a  
 * test failure is signalled.  
 *  
 * @{  
 */  
  
/**  
 * @brief Function to replace assert(3) in tested code.  
 *  
 * In conjunction with check_assert() it's possible to determine whether an  
 * assert condition has failed without stopping a test.  
 *  
 * @param[in] result The expression to assert.  
 *  
 * @param[in] expression The expression as string.  
 *
```

```
* @param[in] file The file mock_assert() is called.  
*  
* @param[in] line The line mock_assert() is called.  
*  
* @code  
* #ifdef UNIT_TESTING  
* extern void mock_assert(const int result, const char* const expression,  
*                         const char * const file, const int line);  
*  
* #undef assert  
* #define assert(expression) \  
*     mock_assert((int)(expression), #expression, __FILE__, __LINE__);  
* #endif  
*  
* void increment_value(int * const value) {  
*     assert(value);  
*     (*value)++;  
* }  
* @endcode  
*  
* @see assert(3)  
* @see expect_assert_failure  
*/  
void mock_assert(const int result, const char* const expression,  
                const char * const file, const int line);
```

```
#ifdef DOXYGEN  
/**  
* @brief Ensure that mock_assert() is called.
```

```
*  
* If mock_assert() is called the assert expression string is returned.  
*  
* @param[in] fn_call The function will will call mock_assert().  
*  
* @code  
* #define assert mock_assert  
*  
* void showmessage(const char *message) {  
*   assert(message);  
* }  
*  
* int main(int argc, const char* argv[]) {  
*   expect_assert_failure(show_message(NULL));  
*   printf("succeeded\n");  
*   return 0;  
* }  
* @endcode  
*  
*/  
void expect_assert_failure(function fn_call);  
#else  
#define expect_assert_failure(function_call) \  
{\ \  
  const int result = setjmp(global_expect_assert_env); \  
  global_expecting_assert = 1; \  
  if (result) { \  
    print_message("Expected assertion %s occurred\n", \  
      global_last_failed_assert); \  
}
```

```

global_expectning_assert = 0; \
} else { \
    function_call ; \
    global_expectning_assert = 0; \
    print_error("Expected assert in %s\n", #function_call); \
    _fail(__FILE__, __LINE__); \
} \
}

#endif

/** @} */

/* Function prototype for setup, test and teardown functions. */
typedef void (*UnitTestFunction)(void **state);

/* Function that determines whether a function parameter value is correct. */
typedef int (*CheckParameterValue)(const LargestIntegralType value,
                                  const LargestIntegralType check_value_data);

/* Type of the unit test function. */
typedef enum UnitTestFunctionType {
    UNIT_TEST_FUNCTION_TYPE_TEST = 0,
    UNIT_TEST_FUNCTION_TYPE_SETUP,
    UNIT_TEST_FUNCTION_TYPE_TEARDOWN,
    UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP,
    UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN,
} UnitTestFunctionType;

/*

```

```
* Stores a unit test function with its name and type.  
* NOTE: Every setup function must be paired with a teardown function. It's  
* possible to specify NULL function pointers.  
*/
```

```
typedef struct UnitTest {  
    const char* name;  
    UnitTestFunction function;  
    UnitTestFunctionType function_type;  
} UnitTest;
```

```
typedef struct GroupTest {  
    UnitTestFunction setup;  
    UnitTestFunction teardown;  
    const UnitTest *tests;  
    const size_t number_of_tests;  
} GroupTest;
```

```
/* Function prototype for test functions. */  
typedef void (*CMUnitTestFunction)(void **state);
```

```
/* Function prototype for setup and teardown functions. */  
typedef int (*CMFixtureFunction)(void **state);
```

```
struct CMUnitTest {  
    const char *name;  
    CMUnitTestFunction test_func;  
    CMFixtureFunction setup_func;  
    CMFixtureFunction teardown_func;  
    void *initial_state;
```

```
};

/* Location within some source code. */
typedef struct SourceLocation {
    const char* file;
    int line;
} SourceLocation;

/* Event that's called to check a parameter value. */
typedef struct CheckParameterEvent {
    SourceLocation location;
    const char *parameter_name;
    CheckParameterValue check_value;
    LargestIntegralType check_value_data;
} CheckParameterEvent;

/* Used by expect_assert_failure() and mock_assert(). */
extern int global_expect_assert;
extern jmp_buf global_expect_assert_env;
extern const char * global_last_failed_assert;

/* Retrieves a value for the given function, as set by "will_return". */
LargestIntegralType _mock(const char * const function, const char* const file,
                           const int line);

void _expect_function_call(
    const char * const function_name,
    const char * const file,
    const int line,
```

```
    const int count);

void _function_called(const char * const function, const char* const file,
                     const int line);

void _expect_check(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const CheckParameterValue check_function,
    const LargestIntegralType check_data, CheckParameterEvent * const event,
    const int count);

void _expect_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType values[],
    const size_t number_of_values, const int count);

void _expect_not_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType values[],
    const size_t number_of_values, const int count);

void _expect_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType minimum,
    const LargestIntegralType maximum, const int count);

void _expect_not_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
```

```
const LargestIntegralType minimum,
const LargestIntegralType maximum, const int count);

void _expect_value(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType value,
    const int count);

void _expect_not_value(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType value,
    const int count);

void _expect_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count);

void _expect_not_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count);

void _expect_memory(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const void* const memory,
    const size_t size, const int count);

void _expect_not_memory(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const void* const memory,
    const size_t size, const int count);
```

```
void _expect_any(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const int count);

void _check_expected(
    const char * const function_name, const char * const parameter_name,
    const char* file, const int line, const LargestIntegralType value);

void _will_return(const char * const function_name, const char * const file,
                  const int line, const LargestIntegralType value,
                  const int count);

void _assert_true(const LargestIntegralType result,
                  const char* const expression,
                  const char * const file, const int line);

void _assert_return_code(const LargestIntegralType result,
                        size_t rlen,
                        const LargestIntegralType error,
                        const char * const expression,
                        const char * const file,
                        const int line);

void _assert_int_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);

void _assert_int_not_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);

void _assert_string_equal(const char * const a, const char * const b,
                        const char * const file, const int line);
```

```
void _assert_string_not_equal(const char * const a, const char * const b,
                             const char *file, const int line);

void _assert_memory_equal(const void * const a, const void * const b,
                        const size_t size, const char* const file,
                        const int line);

void _assert_memory_not_equal(const void * const a, const void * const b,
                            const size_t size, const char* const file,
                            const int line);

void _assert_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);

void _assert_not_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);

void _assert_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line);

void _assert_not_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line);

void* _test_malloc(const size_t size, const char* file, const int line);
void* _test_realloc(void *ptr, const size_t size, const char* file, const int line);
void* _test_calloc(const size_t number_of_elements, const size_t size,
                  const char* file, const int line);

void _test_free(void* const ptr, const char* file, const int line);

void _fail(const char * const file, const int line);
```

```
void _skip(const char * const file, const int line);

int _run_test(
    const char * const function_name, const UnitTestFunction Function,
    void ** const volatile state, const UnitTestFunctionType function_type,
    const void* const heap_check_point);

CMOCKA_DEPRECATED int _run_tests(const UnitTest * const tests,
                                const size_t number_of_tests);

CMOCKA_DEPRECATED int _run_group_tests(const UnitTest * const tests,
                                       const size_t number_of_tests);

/* Test runner */

int _cmocka_run_group_tests(const char *group_name,
                           const struct CMUnitTest * const tests,
                           const size_t num_tests,
                           CMFixtureFunction group_setup,
                           CMFixtureFunction group_teardown);

/* Standard output and error print methods. */

void print_message(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void print_error(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void vprint_message(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);
void vprint_error(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);

enum cm_message_output {
    CM_OUTPUT_STDOUT,
    CM_OUTPUT_SUBUNIT,
    CM_OUTPUT_TAP,
    CM_OUTPUT_XML,
```

```
};

/** 
 * @brief Function to set the output format for a test.
 *
 * The ouput format for the test can either be set globally using this
 * function or overriden with environment variable CMOCKA_MESSAGE_OUTPUT.
 *
 * The environment variable can be set to either STDOUT, SUBUNIT, TAP or XML.
 *
 * @param[in] output  The output format to use for the test.
 *
 */
void cmocka_set_message_output(enum cm_message_output output);

/** @} */

#endif /* CMOCKA_H_ */

#include <cmocka.h>

#include <cmocka_pbc.h>
/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 *
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 */
```

* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.

*/

```
#ifndef CMOCKA_PRIVATE_H_
```

```
#define CMOCKA_PRIVATE_H_
```

```
#ifdef HAVE_CONFIG_H
```

```
#include "config.h"
```

```
#endif
```

```
#include <stdint.h>
```

```
#ifdef _WIN32
```

```
#include <windows.h>
```

```
# ifdef _MSC_VER
```

```
# include <stdio.h> /* __snprintf */
```

```
# undef inline
```

```
# define inline __inline
```

```
# ifndef va_copy
```

```
# define va_copy(dest, src) (dest = src)
```

```
# endif
```

```
# define strcasecmp _stricmp
# define strncasecmp _strnicmp

# if defined(HAVE__SNPRINTF_S)
# undef snprintf
# define snprintf(d, n, ...) _snprintf_s((d), (n), _TRUNCATE, __VA_ARGS__)
# else /* HAVE__SNPRINTF_S */
# if defined(HAVE__SNPRINTF)
# undef snprintf
# define snprintf _snprintf
# else /* HAVE__SNPRINTF */
# if !defined(HAVE__SNPRINTF)
# error "no snprintf compatible function found"
# endif /* HAVE__SNPRINTF */
# endif /* HAVE__SNPRINTF */
# endif /* HAVE__SNPRINTF_S */

# if defined(HAVE__VSNPRINTF_S)
# undef vsnprintf
# define vsnprintf(s, n, f, v) _vsnprintf_s((s), (n), _TRUNCATE, (f), (v))
# else /* HAVE__VSNPRINTF_S */
# if defined(HAVE__VSNPRINTF)
# undef vsnprintf
# define vsnprintf _vsnprintf
# else
# if !defined(HAVE__VSNPRINTF)
# error "No vsnprintf compatible function found"
# endif /* HAVE__VSNPRINTF */
# endif /* HAVE__VSNPRINTF */
# endif /* HAVE__VSNPRINTF_S */
```

```
# endif /* HAVE__VSNPRINTF_S */  
# endif /* _MSC_VER */  
  
/*  
 * Backwards compatibility with headers shipped with Visual Studio 2005 and  
 * earlier.  
 */  
WINBASEAPI BOOL WINAPI IsDebuggerPresent(VOID);  
  
#ifndef PRIdS  
# define PRIdS "Id"  
#endif  
  
#ifndef PRIu64  
# define PRIu64 "I64u"  
#endif  
  
#ifndef PRIuMAX  
# define PRIuMAX PRIu64  
#endif  
  
#ifndef PRIxMAX  
#define PRIxMAX "I64x"  
#endif  
  
#ifndef PRIxMAX  
#define PRIxMAX "I64X"  
#endif
```

```
#else /* _WIN32 */

#ifndef __PRI64_PREFIX
# if __WORDSIZE == 64
# define __PRI64_PREFIX "I"
# else
# define __PRI64_PREFIX "II"
# endif
#endif

#ifndef PRIdS
# define PRIdS "zd"
#endif

#ifndef PRIu64
# define PRIu64 __PRI64_PREFIX "u"
#endif

#ifndef PRIuMAX
# define PRIuMAX __PRI64_PREFIX "u"
#endif

#ifndef PRIxMAX
#define PRIxMAX __PRI64_PREFIX "x"
#endif

#ifndef PRIxMAX
#define PRIxMAX __PRI64_PREFIX "X"
#endif
```

```
#endif /* _WIN32 */\n\n/** Free memory space */\n#define SAFE_FREE(x) do { if ((x) != NULL) {free(x); x=NULL;} } while(0)\n\n/** Zero a structure */\n#define ZERO_STRUCT(x) memset((char *)&(x), 0, sizeof(x))\n\n/** Zero a structure given a pointer to the structure */\n#define ZERO_STRUCTP(x) do { if ((x) != NULL) memset((char *)(x), 0, sizeof(*(x))); } while(0)\n\n/** Get the size of an array */\n#define ARRAY_SIZE(a) (sizeof(a)/sizeof(a[0]))\n\n/** Overwrite the complete string with 'X' */\n#define BURN_STRING(x) do { if ((x) != NULL) memset((x), 'X', strlen((x))); } while(0)\n\n/**\n * This is a hack to fix warnings. The idea is to use this everywhere that we\n * get the "discarding const" warning by the compiler. That doesn't actually\n * fix the real issue, but marks the place and you can search the code for\n * discard_const.\n *\n * Please use this macro only when there is no other way to fix the warning.\n *\n * We should use this function in only in a very few places.\n *\n * Also, please call this via the discard_const_p() macro interface, as that\n * makes the return type safe.\n */
```

```
*/  
#define discard_const(ptr) ((void *)((uintptr_t)(ptr)))  
  
/**  
 * Type-safe version of discard_const  
 */  
#define discard_const_p(type, ptr) ((type *)discard_const(ptr))  
  
#endif /* CMOCKA_PRIVATE_H */  
  
/*  
 * Copyright 2008 Google Inc.  
 * Copyright 2014-2015 Andreas Schneider <asn@cryptomilk.org>  
 * Copyright 2015 Jakub Hrozek <jakub.hrozek@posteo.se>  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 * http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
#ifdef HAVE_CONFIG_H  
#include "config.h"  
#endif
```

```
#ifdef HAVE_MALLOC_H
#include <malloc.h>
#endif

#ifndef HAVE_INNTYPES_H
#include <inttypes.h>
#endif

#ifndef HAVE_SIGNAL_H
#include <signal.h>
#endif

#ifndef HAVE_STRINGS_H
#include <strings.h>
#endif

#include <stdint.h>
#include <setjmp.h>
#include <stdarg.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

/*
 * This allows to add a platform specific header file. Some embedded platforms
 * sometimes miss certain types and definitions.
*/
```

```
*  
* Example:  
*  
* typedef unsigned long int uintptr_t  
* #define _UINTPTR_T 1  
* #define _UINTPTR_T_DEFINED 1  
*/  
  
#ifdef CMOCKA_PLATFORM_INCLUDE  
  
# include "cmocka_platform.h"  
#endif /* CMOCKA_PLATFORM_INCLUDE */  
  
  
#include <cmocka.h>  
#include <cmocka_private.h>  
  
/* Size of guard bytes around dynamically allocated blocks. */  
  
#define MALLOC_GUARD_SIZE 16  
  
/* Pattern used to initialize guard blocks. */  
  
#define MALLOC_GUARD_PATTERN 0xEF  
  
/* Pattern used to initialize memory allocated with test_malloc(). */  
  
#define MALLOC_ALLOC_PATTERN 0xBA  
  
#define MALLOC_FREE_PATTERN 0xCD  
  
/* Alignment of allocated blocks. NOTE: This must be base2. */  
  
#define MALLOC_ALIGNMENT sizeof(size_t)  
  
  
/* Printf formatting for source code locations. */  
  
#define SOURCE_LOCATION_FORMAT "%s:%u"  
  
  
#if defined(HAVE_GCC_THREAD_LOCAL_STORAGE)  
# define CMOCKA_THREAD __thread
```

```
#elif defined(HAVE_MSVC_THREAD_LOCAL_STORAGE)
#define CMOCKA_THREAD __declspec(thread)

#else
#define CMOCKA_THREAD
#endif

#ifndef HAVE_CLOCK_GETTIME_REALTIME
#define CMOCKA_CLOCK_GETTIME(clock_id, ts) clock_gettime((clock_id), (ts))
#else
#define CMOCKA_CLOCK_GETTIME(clock_id, ts)
#endif

#ifndef MAX
#define MAX(a,b) ((a) < (b) ? (b) : (a))
#endif

/***
 * POSIX has sigsetjmp/siglongjmp, while Windows only has setjmp/longjmp.
 */
#ifndef HAVE_SIGLONGJMP
#define cm_jmp_buf      sigjmp_buf
#define cm_setjmp(env)   sigsetjmp(env, 1)
#define cm_longjmp(env, val) siglongjmp(env, val)
#else
#define cm_jmp_buf      jmp_buf
#define cm_setjmp(env)   setjmp(env)
#define cm_longjmp(env, val) longjmp(env, val)
#endif
```

```

/*
 * Declare and initialize the pointer member of ValuePointer variable name
 * with ptr.
 */
#define declare_initialize_value_pointer_pointer(name, ptr) \
    ValuePointer name ; \
    name.value = 0; \
    name.x.pointer = (void*)(ptr)

/*
 * Declare and initialize the value member of ValuePointer variable name
 * with val.
 */
#define declare_initialize_value_pointer_value(name, val) \
    ValuePointer name ; \
    name.value = val

/* Cast a LargestIntegralType to pointer_type via a ValuePointer. */
#define cast_largest_integral_type_to_pointer( \
    pointer_type, largest_integral_type) \
    ((pointer_type)((ValuePointer*)&(largest_integral_type))->x.pointer)

/* Used to cast LargeIntegralType to void* and vice versa. */
typedef union ValuePointer {
    LargestIntegralType value;
    struct {
#endif defined(WORDS_BIGENDIAN) && (WORDS_SIZEOF_VOID_P == 4)
        unsigned int padding;

```

```

#endif

    void *pointer;

} x;

} ValuePointer;

/* Doubly linked list node. */
typedef struct ListNode {

    const void *value;

    int refcount;

    struct ListNode *next;

    struct ListNode *prev;

} ListNode;

/* Debug information for malloc(). */
typedef struct MallocBlockInfo {

    void* block;          /* Address of the block returned by malloc(). */

    size_t allocated_size; /* Total size of the allocated block. */

    size_t size;           /* Request block size. */

    SourceLocation location; /* Where the block was allocated. */

    ListNode node;         /* Node within list of all allocated blocks. */

} MallocBlockInfo;

/* State of each test. */
typedef struct TestState {

    const ListNode *check_point; /* Check point of the test if there's a */

                                /* setup function. */

    void *state;              /* State associated with the test. */

} TestState;

```

```

/* Determines whether two values are the same. */
typedef int (*EqualityFunction)(const void *left, const void *right);

/* Value of a symbol and the place it was declared. */
typedef struct SymbolValue {
    SourceLocation location;
    LargestIntegralType value;
} SymbolValue;

/*
 * Contains a list of values for a symbol.
 * NOTE: Each structure referenced by symbol_values_list_head must have a
 * SourceLocation as its' first member.
 */
typedef struct SymbolMapView {
    const char *symbol_name;
    ListNode symbol_values_list_head;
} SymbolMapView;

/* Where a particular ordering was located and its symbol name */
typedef struct FuncOrderingValue {
    SourceLocation location;
    const char * function;
} FuncOrderingValue;

/* Used by list_free() to deallocate values referenced by list nodes. */
typedef void (*CleanupListValue)(const void *value, void *cleanup_value_data);

/* Structure used to check the range of integer types.a */

```

```

typedef struct CheckIntegerRange {
    CheckParameterEvent event;
    LargestIntegralType minimum;
    LargestIntegralType maximum;
} CheckIntegerRange;

/* Structure used to check whether an integer value is in a set. */

typedef struct CheckIntegerSet {
    CheckParameterEvent event;
    const LargestIntegralType *set;
    size_t size_of_set;
} CheckIntegerSet;

/* Used to check whether a parameter matches the area of memory referenced by
 * this structure. */

typedef struct CheckMemoryData {
    CheckParameterEvent event;
    const void *memory;
    size_t size;
} CheckMemoryData;

static ListNode* list_initialize(ListNode * const node);

static ListNode* list_add(ListNode * const head, ListNode *new_node);

static ListNode* list_add_value(ListNode * const head, const void *value,
                               const int count);

static ListNode* list_remove(
    ListNode * const node, const CleanupListValue cleanup_value,
    void * const cleanup_value_data);

static void list_remove_free(

```

```
ListNode * const node, const CleanupListValue cleanup_value,
void * const cleanup_value_data);

static int list_empty(const ListNode * const head);

static int list_find(
    ListNode * const head, const void *value,
    const EqualityFunction equal_func, ListNode **output);

static int list_first(ListNode * const head, ListNode **output);

static ListNode* list_free(
    ListNode * const head, const CleanupListValue cleanup_value,
    void * const cleanup_value_data);

static void add_symbol_value(
    ListNode * const symbol_map_head, const char * const symbol_names[],
    const size_t number_of_symbol_names, const void* value, const int count);

static int get_symbol_value(
    ListNode * const symbol_map_head, const char * const symbol_names[],
    const size_t number_of_symbol_names, void **output);

static void free_value(const void *value, void *cleanup_value_data);

static void free_symbol_map_value(
    const void *value, void *cleanup_value_data);

static void remove_always_return_values(ListNode * const map_head,
    const size_t number_of_symbol_names);

static int check_for_leftover_values_list(const ListNode * head,
    const char * const error_message);

static int check_for_leftover_values(
    const ListNode * const map_head, const char * const error_message,
    const size_t number_of_symbol_names);
```

```
static void remove_always_return_values_from_list(ListNode * const map_head);
```

```
/*
```

```
* This must be called at the beginning of a test to initialize some data  
* structures.
```

```
*/
```

```
static void initialize_testing(const char *test_name);
```

```
/* This must be called at the end of a test to free() allocated structures. */
```

```
static void teardown_testing(const char *test_name);
```

```
static enum cm_message_output cm_get_output(void);
```

```
static int cm_error_message_enabled = 1;
```

```
static CMOCKA_THREAD char *cm_error_message;
```

```
void cm_print_error(const char * const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
```

```
/*
```

```
* Keeps track of the calling context returned by setenv() so that the fail()  
* method can jump out of a test.
```

```
*/
```

```
static CMOCKA_THREAD cm_jmp_buf global_run_test_env;
```

```
static CMOCKA_THREAD int global_running_test = 0;
```

```
/* Keeps track of the calling context returned by setenv() so that */
```

```
/* mock_assert() can optionally jump back to expect_assert_failure(). */
```

```
jmp_buf global_expect_assert_env;
```

```
int global_expectng_assert = 0;

const char *global_last_failed_assert = NULL;

static int global_skip_test;

/* Keeps a map of the values that functions will have to return to provide */

/* mocked interfaces. */

static CMOCKA_THREAD ListNode global_function_result_map_head;

/* Location of the last mock value returned was declared. */

static CMOCKA_THREAD SourceLocation global_last_mock_value_location;

/* Keeps a map of the values that functions expect as parameters to their
 * mocked interfaces. */

static CMOCKA_THREAD ListNode global_function_parameter_map_head;

/* Location of last parameter value checked was declared. */

static CMOCKA_THREAD SourceLocation global_last_parameter_location;

/* List (acting as FIFO) of call ordering. */

static CMOCKA_THREAD ListNode global_call_ordering_head;

/* Location of last call ordering that was declared. */

static CMOCKA_THREAD SourceLocation global_last_call_ordering_location;

/* List of all currently allocated blocks. */

static CMOCKA_THREAD ListNode global_allocated_blocks;

static enum cm_message_output global_msg_output = CM_OUTPUT_STDOUT;

#ifndef _WIN32

/* Signals caught by exception_handler(). */

static const int exception_signals[] = {
```

```
SIGFPE,  
SIGILL,  
SIGSEGV,  
#ifdef SIGBUS  
    SIGBUS,  
#endif  
#ifdef SIGSYS  
    SIGSYS,  
#endif  
};  
  
/* Default signal functions that should be restored after a test is complete. */  
typedef void (*SignalFunction)(int signal);  
static SignalFunction default_signal_functions[  
    ARRAY_SIZE(exception_signals)];  
  
#else /* _WIN32 */  
  
/* The default exception filter. */  
static LPTOP_LEVEL_EXCEPTION_FILTER previous_exception_filter;  
  
/* Fatal exceptions. */  
typedef struct ExceptionCodeInfo {  
    DWORD code;  
    const char* description;  
} ExceptionCodeInfo;  
  
#define EXCEPTION_CODE_INFO(exception_code) {exception_code, #exception_code}
```

```
static const ExceptionCodeInfo exception_codes[] = {  
    EXCEPTION_CODE_INFO(EXCEPTION_ACCESS_VIOLATION),  
    EXCEPTION_CODE_INFO(EXCEPTION_ARRAY_BOUNDS_EXCEEDED),  
    EXCEPTION_CODE_INFO(EXCEPTION_DATATYPE_MISALIGNMENT),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_DENORMAL_OPERAND),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_DIVIDE_BY_ZERO),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_INEXACT_RESULT),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_INVALID_OPERATION),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_OVERFLOW),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_STACK_CHECK),  
    EXCEPTION_CODE_INFO(EXCEPTION_FLT_UNDERFLOW),  
    EXCEPTION_CODE_INFO(EXCEPTION_GUARD_PAGE),  
    EXCEPTION_CODE_INFO(EXCEPTION_ILLEGAL_INSTRUCTION),  
    EXCEPTION_CODE_INFO(EXCEPTION_INT_DIVIDE_BY_ZERO),  
    EXCEPTION_CODE_INFO(EXCEPTION_INT_OVERFLOW),  
    EXCEPTION_CODE_INFO(EXCEPTION_INVALID_DISPOSITION),  
    EXCEPTION_CODE_INFO(EXCEPTION_INVALID_HANDLE),  
    EXCEPTION_CODE_INFO(EXCEPTION_IN_PAGE_ERROR),  
    EXCEPTION_CODE_INFO(EXCEPTION_NONCONTINUABLE_EXCEPTION),  
    EXCEPTION_CODE_INFO(EXCEPTION_PRIV_INSTRUCTION),  
    EXCEPTION_CODE_INFO(EXCEPTION_STACK_OVERFLOW),  
};  
#endif /* !_WIN32 */
```

```
enum CMUnitTestStatus {  
    CM_TEST_NOT_STARTED,  
    CM_TEST_PASSED,  
    CM_TEST_FAILED,  
    CM_TEST_ERROR,
```

```

CM_TEST_SKIPPED,
};

struct CMUnitTestState {
    const ListNode *check_point; /* Check point of the test if there's a setup function. */
    const struct CMUnitTest *test; /* Point to array element in the tests we get passed */
    void *state; /* State associated with the test */
    const char *error_message; /* The error messages by the test */
    enum CMUnitTestStatus status; /* PASSED, FAILED, ABORT ... */
    double runtime; /* Time calculations */
};

/* Exit the currently executing test. */
static void exit_test(const int quit_application)
{
    const char *abort_test = getenv("CMOCKA_TEST_ABORT");

    if (abort_test != NULL && abort_test[0] == '1') {
        print_error("%s", cm_error_message);
        abort();
    } else if (global_running_test) {
        cm_longjmp(global_run_test_env, 1);
    } else if (quit_application) {
        exit(-1);
    }
}

void _skip(const char * const file, const int line)
{

```

```
cm_print_error(SOURCE_LOCATION_FORMAT ": Skipped!\n", file, line);
global_skip_test = 1;
exit_test(1);

}

/* Initialize a SourceLocation structure.*/
static void initialize_source_location(SourceLocation * const location) {
    assert_non_null(location);
    location->file = NULL;
    location->line = 0;
}

/* Determine whether a source location is currently set.*/
static int source_location_is_set(const SourceLocation * const location) {
    assert_non_null(location);
    return location->file && location->line;
}

/* Set a source location.*/
static void set_source_location(
    SourceLocation * const location, const char * const file,
    const int line) {
    assert_non_null(location);
    location->file = file;
    location->line = line;
}
```

```
static int c_strreplace(char *src,
    size_t src_len,
    const char *pattern,
    const char *repl,
    int *str_replaced)

{
    char *p = NULL;

    p = strstr(src, pattern);
    if (p == NULL) {
        return -1;
    }

    do {
        size_t of = p - src;
        size_t l = strlen(src);
        size_t pl = strlen(pattern);
        size_t rl = strlen(repl);

        /* overflow check */
        if (src_len <= l + MAX(pl, rl) + 1) {
            return -1;
        }

        if (rl != pl) {
            memmove(src + of + rl, src + of + pl, l - of - pl + 1);
        }
    }
```

```

strncpy(src + of, repl, rl);

if (str_replaced != NULL) {
    *str_replaced = 1;
}

p = strstr(src, pattern);

} while (p != NULL);

return 0;
}

/* Create function results and expected parameter lists. */
void initialize_testing(const char *test_name) {
    (void)test_name;

    list_initialize(&global_function_result_map_head);

    initialize_source_location(&global_last_mock_value_location);

    list_initialize(&global_function_parameter_map_head);

    initialize_source_location(&global_last_parameter_location);

    list_initialize(&global_call_ordering_head);

    initialize_source_location(&global_last_parameter_location);

}
}

static void fail_if_leftover_values(const char *test_name) {
    int error_occurred = 0;

    (void)test_name;

    remove_always_return_values(&global_function_result_map_head, 1);

    if (check_for_leftover_values(
        &global_function_result_map_head,

```

```
%s() has remaining non-returned values.\n", 1)) {  
    error_occurred = 1;  
}  
  
remove_always_return_values(&global_function_parameter_map_head, 2);  
if (check_for_leftover_values(  
    &global_function_parameter_map_head,  
    "%s parameter still has values that haven't been checked.\n", 2)) {  
    error_occurred = 1;  
}  
  
remove_always_return_values_from_list(&global_call_ordering_head);  
if (check_for_leftover_values_list(&global_call_ordering_head,  
    "%s function was expected to be called but was not not.\n")) {  
    error_occurred = 1;  
}  
if (error_occurred) {  
    exit_test(1);  
}  
}  
  
static void teardown_testing(const char *test_name) {  
    (void)test_name;  
    list_free(&global_function_result_map_head, free_symbol_map_value,  
        (void*)0);  
    initialize_source_location(&global_last_mock_value_location);  
    list_free(&global_function_parameter_map_head, free_symbol_map_value,  
        (void*)1);
```

```
initialize_source_location(&global_last_parameter_location);

list_free(&global_call_ordering_head, free_value,
          (void*)0);

initialize_source_location(&global_last_call_ordering_location);

}
```

```
/* Initialize a list node. */

static ListNode* list_initialize(ListNode * const node) {

    node->value = NULL;

    node->next = node;

    node->prev = node;

    node->refcount = 1;

    return node;

}
```

```
/*
 * Adds a value at the tail of a given list.
 *
 * The node referencing the value is allocated from the heap.
 */

static ListNode* list_add_value(ListNode * const head, const void *value,
                               const int refcount) {

    ListNode * const new_node = (ListNode*)malloc(sizeof(ListNode));

    assert_non_null(head);

    assert_non_null(value);

    new_node->value = value;

    new_node->refcount = refcount;

    return list_add(head, new_node);

}
```

```
/* Add new_node to the end of the list. */

static ListNode* list_add(ListNode * const head, ListNode *new_node) {

    assert_non_null(head);

    assert_non_null(new_node);

    new_node->next = head;

    new_node->prev = head->prev;

    head->prev->next = new_node;

    head->prev = new_node;

    return new_node;

}
```

```
/* Remove a node from a list. */

static ListNode* list_remove(

    ListNode * const node, const CleanupListValue cleanup_value,

    void * const cleanup_value_data) {

    assert_non_null(node);

    node->prev->next = node->next;

    node->next->prev = node->prev;

    if (cleanup_value) {

        cleanup_value(node->value, cleanup_value_data);

    }

    return node;

}
```

```
/* Remove a list node from a list and free the node. */
```

```

static void list_remove_free(
    ListNode * const node, const CleanupListValue cleanup_value,
    void * const cleanup_value_data) {
    assert_non_null(node);
    free(list_remove(node, cleanup_value, cleanup_value_data));
}

/*
 * Frees memory kept by a linked list. The cleanup_value function is called for
 * every "value" field of nodes in the list, except for the head. In addition
 * to each list value, cleanup_value_data is passed to each call to
 * cleanup_value. The head of the list is not deallocated.
 */
static ListNode* list_free(
    ListNode * const head, const CleanupListValue cleanup_value,
    void * const cleanup_value_data) {
    assert_non_null(head);
    while (!list_empty(head)) {
        list_remove_free(head->next, cleanup_value, cleanup_value_data);
    }
    return head;
}

/* Determine whether a list is empty. */
static int list_empty(const ListNode * const head) {
    assert_non_null(head);
    return head->next == head;
}

```

```
}

/*
 * Find a value in the list using the equal_func to compare each node with the
 * value.
 */

static int list_find(ListNode * const head, const void *value,
                     const EqualityFunction equal_func, ListNode **output) {
    ListNode *current;
    assert_non_null(head);
    for (current = head->next; current != head; current = current->next) {
        if (equal_func(current->value, value)) {
            *output = current;
            return 1;
        }
    }
    return 0;
}

/* Returns the first node of a list */

static int list_first(ListNode * const head, ListNode **output) {
    ListNode *target_node;
    assert_non_null(head);
    if (list_empty(head)) {
        return 0;
    }
    target_node = head->next;
    *output = target_node;
```

```

    return 1;
}

/* Deallocation a value referenced by a list. */

static void free_value(const void *value, void *cleanup_value_data) {
    (void)cleanup_value_data;
    assert_non_null(value);
    free((void*)value);
}

/* Releases memory associated to a symbol_map_value. */

static void free_symbol_map_value(const void *value,
                                  void *cleanup_value_data) {
    SymbolMapValue * const map_value = (SymbolMapValue*)value;
    const LargestIntegralType children = cast_ptr_to_largest_integral_type(cleanup_value_data);
    assert_non_null(value);
    list_free(&map_value->symbol_values_list_head,
              children ? free_symbol_map_value : free_value,
              (void *) ((uintptr_t)children - 1));
    free(map_value);
}

/*
 * Determine whether a symbol name referenced by a symbol_map_value matches the
 * specified function name.
 */

```

```

static int symbol_names_match(const void *map_value, const void *symbol) {
    return !strcmp(((SymbolMapValue*)map_value)->symbol_name,
        (const char*)symbol);
}

/*
 * Adds a value to the queue of values associated with the given hierarchy of
 * symbols. It's assumed value is allocated from the heap.
 */
static void add_symbol_value(ListNode * const symbol_map_head,
    const char * const symbol_names[],
    const size_t number_of_symbol_names,
    const void* value, const int refcount) {
    const char* symbol_name;
    ListNode *target_node;
    SymbolMapView *target_map_value;
    assert_non_null(symbol_map_head);
    assert_non_null(symbol_names);
    assert_true(number_of_symbol_names);
    symbol_name = symbol_names[0];

    if (!list_find(symbol_map_head, symbol_name, symbol_names_match,
        &target_node)) {
        SymbolMapView * const new_symbol_map_value =
            (SymbolMapView*)malloc(sizeof(*new_symbol_map_value));
        new_symbol_map_value->symbol_name = symbol_name;
        list_initialize(&new_symbol_map_value->symbol_values_list_head);
        target_node = list_add_value(symbol_map_head, new_symbol_map_value,
            1);
    }
}

```

```
}

target_map_value = (SymbolMapValue*)target_node->value;

if (number_of_symbol_names == 1) {

    list_add_value(&target_map_value->symbol_values_list_head,
                  value, refcount);

} else {

    add_symbol_value(&target_map_value->symbol_values_list_head,
                     &symbol_names[1], number_of_symbol_names - 1, value,
                     refcount);

}

}
```

```
/*
 * Gets the next value associated with the given hierarchy of symbols.
 * The value is returned as an output parameter with the function returning the
 * node's old refcount value if a value is found, 0 otherwise. This means that
 * a return value of 1 indicates the node was just removed from the list.
 */
```

```
static int get_symbol_value(

    ListNode * const head, const char * const symbol_names[],
    const size_t number_of_symbol_names, void **output) {

    const char* symbol_name;

    ListNode *target_node;

    assert_non_null(head);
    assert_non_null(symbol_names);
    assert_true(number_of_symbol_names);
    assert_non_null(output);
```

```

symbol_name = symbol_names[0];

if (list_find(head, symbol_name, symbol_names_match, &target_node)) {
    SymbolMapValue *map_value;
    ListNode *child_list;
    int return_value = 0;
    assert_non_null(target_node);
    assert_non_null(target_node->value);

    map_value = (SymbolMapValue*)target_node->value;
    child_list = &map_value->symbol_values_list_head;

    if (number_of_symbol_names == 1) {
        ListNode *value_node = NULL;
        return_value = list_first(child_list, &value_node);
        assert_true(return_value);
        *output = (void*) value_node->value;
        return_value = value_node->refcount;
        if (value_node->refcount - 1 == 0) {
            list_remove_free(value_node, NULL, NULL);
        } else if (value_node->refcount > WILL_RETURN_ONCE) {
            --value_node->refcount;
        }
    } else {
        return_value = get_symbol_value(
            child_list, &symbol_names[1], number_of_symbol_names - 1,
            output);
    }
    if (list_empty(child_list)) {

```

```

        list_remove_free(target_node, free_symbol_map_value, (void*)0);

    }

    return return_value;
} else {
    cm_print_error("No entries for symbol %s.\n", symbol_name);
}

return 0;
}

/** 
 * Traverse a list of nodes and remove first symbol value in list that has a
 * refcount < -1 (i.e. should always be returned and has been returned at
 * least once).
*/

```

```

static void remove_always_return_values_from_list(ListNode * const map_head)
{
    ListNode * current = NULL;
    ListNode * next = NULL;
    assert_non_null(map_head);

    for (current = map_head->next, next = current->next;
         current != map_head;
         current = next, next = current->next) {
        if (current->refcount < -1) {
            list_remove_free(current, free_value, NULL);
        }
    }
}

```

```

/*
 * Traverse down a tree of symbol values and remove the first symbol value
 * in each branch that has a refcount < -1 (i.e should always be returned
 * and has been returned at least once).
 */

static void remove_always_return_values(ListNode * const map_head,
                                         const size_t number_of_symbol_names) {
    ListNode *current;
    assert_non_null(map_head);
    assert_true(number_of_symbol_names);
    current = map_head->next;
    while (current != map_head) {
        SymbolMapValue * const value = (SymbolMapValue*)current->value;
        ListNode * const next = current->next;
        ListNode *child_list;
        assert_non_null(value);
        child_list = &value->symbol_values_list_head;

        if (!list_empty(child_list)) {
            if (number_of_symbol_names == 1) {
                ListNode * const child_node = child_list->next;
                /* If this item has been returned more than once, free it. */
                if (child_node->refcount < -1) {
                    list_remove_free(child_node, free_value, NULL);
                }
            } else {
                remove_always_return_values(child_list,
                                            number_of_symbol_names - 1);
            }
        }
    }
}

```

```

        }

    }

    if (list_empty(child_list)) {
        list_remove_free(current, free_value, NULL);
    }
    current = next;
}

static int check_for_leftover_values_list(const ListNode * head,
                                         const char * const error_message)
{
    ListNode *child_node;
    int leftover_count = 0;
    if (!list_empty(head))
    {
        for (child_node = head->next; child_node != head;
             child_node = child_node->next, ++leftover_count) {
            const FuncOrderingValue *const o =
                (const FuncOrderingValue*) child_node->value;
            cm_print_error(error_message, o->function);
            cm_print_error(SOURCE_LOCATION_FORMAT
                           ": note: remaining item was declared here\n",
                           o->location.file, o->location.line);
        }
    }
    return leftover_count;
}

```

```

/*
 * Checks if there are any leftover values set up by the test that were never
 * retrieved through execution, and fail the test if that is the case.
 */

static int check_for_leftover_values(
    const ListNode * const map_head, const char * const error_message,
    const size_t number_of_symbol_names) {
    const ListNode *current;
    int symbols_with_leftover_values = 0;
    assert_non_null(map_head);
    assert_true(number_of_symbol_names);

    for (current = map_head->next; current != map_head;
         current = current->next) {
        const SymbolMapValue * const value =
            (SymbolMapValue*)current->value;
        const ListNode *child_list;
        assert_non_null(value);
        child_list = &value->symbol_values_list_head;

        if (!list_empty(child_list)) {
            if (number_of_symbol_names == 1) {
                const ListNode *child_node;
                cm_print_error(error_message, value->symbol_name);

                for (child_node = child_list->next; child_node != child_list;
                     child_node = child_node->next) {
                    const SourceLocation * const location =

```

```

        (const SourceLocation*)child_node->value;

        cm_print_error(SOURCE_LOCATION_FORMAT
                      ": note: remaining item was declared here\n",
                      location->file, location->line);

    }

} else {
    cm_print_error("%s.", value->symbol_name);
    check_for_leftover_values(child_list, error_message,
                              number_of_symbol_names - 1);
}

symbols_with_leftover_values++;

}

return symbols_with_leftover_values;
}

```

```

/* Get the next return value for the specified mock function. */

LargestIntegralType _mock(const char * const function, const char* const file,
                         const int line) {

void *result;

const int rc = get_symbol_value(&global_function_result_map_head,
                               &function, 1, &result);

if (rc) {
    SymbolValue * const symbol = (SymbolValue*)result;

    const LargestIntegralType value = symbol->value;
    global_last_mock_value_location = symbol->location;

    if (rc == 1) {
        free(symbol);
    }
}

```

```

    }

    return value;

} else {

    cm_print_error(SOURCE_LOCATION_FORMAT ": error: Could not get value "
                  "to mock function %s\n", file, line, function);

    if (source_location_is_set(&global_last_mock_value_location)) {

        cm_print_error(SOURCE_LOCATION_FORMAT
                      ": note: Previously returned mock value was declared here\n",
                      global_last_mock_value_location.file,
                      global_last_mock_value_location.line);

    } else {

        cm_print_error("There were no previously returned mock values for "
                      "this test.\n");

    }

    exit_test(1);

}

return 0;
}

```

```

/* Ensure that function is being called in proper order */

void _function_called(const char *const function,
                      const char *const file,
                      const int line)

{
    ListNode *first_value_node = NULL;

    ListNode *value_node = NULL;
    FuncOrderingValue *expected_call;

    int rc;

```

```

rc = list_first(&global_call_ordering_head, &value_node);

first_value_node = value_node;

if (rc) {

    int cmp;

    expected_call = (FuncOrderingValue *)value_node->value;

    cmp = strcmp(expected_call->function, function);

    if (value_node->refcount < -1) {

        /*
         * Search through value nodes until either function is found or
         * encounter a non-zero refcount greater than -2
         */

        if (cmp != 0) {

            value_node = value_node->next;

            expected_call = (FuncOrderingValue *)value_node->value;

            cmp = strcmp(expected_call->function, function);

            while (value_node->refcount < -1 &&
                  cmp != 0 &&
                  value_node != first_value_node->prev) {

                value_node = value_node->next;

                if (value_node == NULL) {

                    break;
                }
            }

            expected_call = (FuncOrderingValue *)value_node->value;

            if (expected_call == NULL) {

                continue;
            }
        }

        cmp = strcmp(expected_call->function, function);
    }
}

```

```
    }

    if (value_node == first_value_node->prev) {
        cm_print_error(SOURCE_LOCATION_FORMAT
                      ": error: No expected mock calls matching "
                      "called() invocation in %s",
                      file, line,
                      function);

        exit_test(1);
    }
}

}

if (cmp == 0) {
    if (value_node->refcount > -2 && --value_node->refcount == 0) {
        list_remove_free(value_node, free_value, NULL);
    }
} else {
    cm_print_error(SOURCE_LOCATION_FORMAT
                  ": error: Expected call to %s but received called() "
                  "in %s\n",
                  file, line,
                  expected_call->function,
                  function);

    exit_test(1);
}
} else {

    cm_print_error(SOURCE_LOCATION_FORMAT
                  ": error: No mock calls expected but called() was "
```

```

        "invoked in %s\n",
        file, line,
        function);
    exit_test(1);
}

}

/* Add a return value for the specified mock function name. */
void _will_return(const char * const function_name, const char * const file,
                  const int line, const LargestIntegralType value,
                  const int count) {
    SymbolValue * const return_value =
        (SymbolValue*)malloc(sizeof(*return_value));
    assert_true(count != 0);
    return_value->value = value;
    set_source_location(&return_value->location, file, line);
    add_symbol_value(&global_function_result_map_head, &function_name, 1,
                     return_value, count);
}

/*
 * Add a custom parameter checking function. If the event parameter is NULL
 * the event structure is allocated internally by this function. If event
 * parameter is provided it must be allocated on the heap and doesn't need to
 * be deallocated by the caller.
 */
void _expect_check(
    const char* const function, const char* const parameter,

```

```
const char* const file, const int line,
const CheckParameterValue check_function,
const LargestIntegralType check_data,
CheckParameterEvent * const event, const int count) {

CheckParameterEvent * const check =
    event ? event : (CheckParameterEvent*)malloc(sizeof(*check));
const char* symbols[] = {function, parameter};
check->parameter_name = parameter;
check->check_value = check_function;
check->check_value_data = check_data;
set_source_location(&check->location, file, line);
add_symbol_value(&global_function_parameter_map_head, symbols, 2, check,
    count);
}
```

```
/*
 * Add an call expectations that a particular function is called correctly.
 * This is used for code under test that makes calls to several functions
 * in depended upon components (mocks).
 */
```

```
void _expect_function_call(
    const char * const function_name,
    const char * const file,
    const int line,
    const int count)

{
    FuncOrderingValue *ordering;
```

```
assert_non_null(function_name);
assert_non_null(file);
assert_true(count != 0);

ordering = (FuncOrderingValue *)malloc(sizeof(*ordering));

set_source_location(&ordering->location, file, line);
ordering->function = function_name;

list_add_value(&global_call_ordering_head, ordering, count);
}

/* Returns 1 if the specified values are equal. If the values are not equal
 * an error is displayed and 0 is returned. */
static int values_equal_display_error(const LargestIntegralType left,
                                      const LargestIntegralType right) {
    const int equal = left == right;
    if (!equal) {
        cm_print_error(LargestIntegralTypePrintfFormat " != "
                      LargestIntegralTypePrintfFormat "\n", left, right);
    }
    return equal;
}

/*
 * Returns 1 if the specified values are not equal. If the values are equal
 * an error is displayed and 0 is returned. */
static int values_not_equal_display_error(const LargestIntegralType left,
```

```

const int not_equal = left != right;

if (!not_equal) {
    cm_print_error(LargestIntegralTypePrintfFormat " == "
                  LargestIntegralTypePrintfFormat "\n", left, right);
}

return not_equal;
}

/*
 * Determine whether value is contained within check_integer_set.
 * If invert is 0 and the value is in the set 1 is returned, otherwise 0 is
 * returned and an error is displayed. If invert is 1 and the value is not
 * in the set 1 is returned, otherwise 0 is returned and an error is
 * displayed.
*/
static int value_in_set_display_error(
    const LargestIntegralType value,
    const CheckIntegerSet * const check_integer_set, const int invert) {
    int succeeded = invert;
    assert_non_null(check_integer_set);
    {
        const LargestIntegralType * const set = check_integer_set->set;
        const size_t size_of_set = check_integer_set->size_of_set;
        size_t i;
        for (i = 0; i < size_of_set; i++) {
            if (set[i] == value) {
                /* If invert = 0 and item is found, succeeded = 1. */
                /* If invert = 1 and item is found, succeeded = 0. */

```

```

        succeeded = !succeeded;
        break;
    }
}

if (succeeded) {
    return 1;
}

cm_print_error(LargestIntegralTypePrintfFormatDecimal
    " is %sin the set (",
    value, invert ? "" : "not ");

for (i = 0; i < size_of_set; i++) {
    cm_print_error(LargestIntegralTypePrintfFormat ", ", set[i]);
}

cm_print_error(")\n");

}
return 0;
}

/*
 * Determine whether a value is within the specified range. If the value is
 * within the specified range 1 is returned. If the value isn't within the
 * specified range an error is displayed and 0 is returned.
 */
static int integer_in_range_display_error(
    const LargestIntegralType value, const LargestIntegralType range_min,
    const LargestIntegralType range_max) {
    if (value >= range_min && value <= range_max) {
        return 1;
    }
}
```

```

}

cm_print_error(LargestIntegralTypePrintfFormatDecimal
    " is not within the range "
    LargestIntegralTypePrintfFormatDecimal "-"
    LargestIntegralTypePrintfFormatDecimal "\n",
    value, range_min, range_max);

return 0;
}

/*
* Determine whether a value is within the specified range. If the value
* is not within the range 1 is returned. If the value is within the
* specified range an error is displayed and zero is returned.
*/
static int integer_not_in_range_display_error(
    const LargestIntegralType value, const LargestIntegralType range_min,
    const LargestIntegralType range_max) {
    if (value < range_min || value > range_max) {
        return 1;
    }
    cm_print_error(LargestIntegralTypePrintfFormatDecimal
        " is within the range "
        LargestIntegralTypePrintfFormatDecimal "-"
        LargestIntegralTypePrintfFormatDecimal "\n",
        value, range_min, range_max);

    return 0;
}

```

```
/*
 * Determine whether the specified strings are equal. If the strings are equal
 * 1 is returned. If they're not equal an error is displayed and 0 is
 * returned.
 */
static int string_equal_display_error(
    const char * const left, const char * const right) {
    if (strcmp(left, right) == 0) {
        return 1;
    }
    cm_print_error("\"%s\" != \"%s\"\n", left, right);
    return 0;
}
```

```
/*
 * Determine whether the specified strings are equal. If the strings are not
 * equal 1 is returned. If they're not equal an error is displayed and 0 is
 * returned
 */
static int string_not_equal_display_error(
    const char * const left, const char * const right) {
    if (strcmp(left, right) != 0) {
        return 1;
    }
    cm_print_error("\"%s\" == \"%s\"\n", left, right);
    return 0;
}
```

```
/*
 * Determine whether the specified areas of memory are equal. If they're equal
 * 1 is returned otherwise an error is displayed and 0 is returned.
 */

static int memory_equal_display_error(const char* const a, const char* const b,
                                      const size_t size) {
    int differences = 0;
    size_t i;
    for (i = 0; i < size; i++) {
        const char l = a[i];
        const char r = b[i];
        if (l != r) {
            cm_print_error("difference at offset %" PRIdS " 0x%02x 0x%02x\n",
                           i, l, r);
            differences++;
        }
    }
    if (differences) {
        cm_print_error("%d bytes of %p and %p differ\n",
                      differences, (void *)a, (void *)b);
        return 0;
    }
    return 1;
}

/*
```

```

* Determine whether the specified areas of memory are not equal. If they're
* not equal 1 is returned otherwise an error is displayed and 0 is
* returned.

*/
static int memory_not_equal_display_error(
    const char* const a, const char* const b, const size_t size) {
    size_t same = 0;
    size_t i;
    for (i = 0; i < size; i++) {
        const char l = a[i];
        const char r = b[i];
        if (l == r) {
            same++;
        }
    }
    if (same == size) {
        cm_print_error("%"PRIIdS "bytes of %p and %p the same\n",
                      same, (void *)a, (void *)b);
        return 0;
    }
    return 1;
}

/* CheckParameterValue callback to check whether a value is within a set. */

static int check_in_set(const LargestIntegralType value,
                       const LargestIntegralType check_value_data) {
    return value_in_set_display_error(value,
                                      cast_largest_integral_type_to_pointer(CheckIntegerSet*,

```

```

        check_value_data), 0);

}

/* CheckParameterValue callback to check whether a value isn't within a set. */

static int check_not_in_set(const LargestIntegralType value,
                           const LargestIntegralType check_value_data) {
    return value_in_set_display_error(value,
                                      cast_largest_integral_type_to_pointer(CheckIntegerSet*,
                                      check_value_data), 1);
}

/* Create the callback data for check_in_set() or check_not_in_set() and
 * register a check event. */

static void expect_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType values[], const size_t number_of_values,
    const CheckParameterValue check_function, const int count) {

    CheckIntegerSet * const check_integer_set =
        (CheckIntegerSet*)malloc(sizeof(*check_integer_set) +
        (sizeof(values[0]) * number_of_values));

    LargestIntegralType * const set = (LargestIntegralType*|(
        check_integer_set + 1);

    declare_initialize_value_pointer_pointer(check_data, check_integer_set);
    assert_non_null(values);
    assert_true(number_of_values);
    memcpy(set, values, number_of_values * sizeof(values[0]));
}

```

```
    check_integer_set->set = set;
    check_integer_set->size_of_set = number_of_values;
    _expect_check(
        function, parameter, file, line, check_function,
        check_data.value, &check_integer_set->event, count);
}
```

```
/* Add an event to check whether a value is in a set. */

void _expect_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType values[], const size_t number_of_values,
    const int count) {

    expect_set(function, parameter, file, line, values, number_of_values,
               check_in_set, count);
}
```

```
/* Add an event to check whether a value isn't in a set. */

void _expect_not_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType values[], const size_t number_of_values,
    const int count) {

    expect_set(function, parameter, file, line, values, number_of_values,
               check_not_in_set, count);
}
```

```
/* CheckParameterValue callback to check whether a value is within a range. */

static int check_in_range(const LargestIntegralType value,
                         const LargestIntegralType check_value_data) {

    CheckIntegerRange * const check_integer_range =
        cast_largest_integral_type_to_pointer(CheckIntegerRange *,
                                              check_value_data);

    assert_non_null(check_integer_range);

    return integer_in_range_display_error(value, check_integer_range->minimum,
                                           check_integer_range->maximum);
}
```

```
/* CheckParameterValue callback to check whether a value is not within a range. */

static int check_not_in_range(const LargestIntegralType value,
                             const LargestIntegralType check_value_data) {

    CheckIntegerRange * const check_integer_range =
        cast_largest_integral_type_to_pointer(CheckIntegerRange *,
                                              check_value_data);

    assert_non_null(check_integer_range);

    return integer_not_in_range_display_error(
        value, check_integer_range->minimum, check_integer_range->maximum);
}
```

```
/* Create the callback data for check_in_range() or check_not_in_range() and
 * register a check event. */

static void expect_range(
    const char* const function, const char* const parameter,
```

```
const char* const file, const int line,
const LargestIntegralType minimum, const LargestIntegralType maximum,
const CheckParameterValue check_function, const int count) {

CheckIntegerRange * const check_integer_range =
    (CheckIntegerRange*)malloc(sizeof(*check_integer_range));

declare_initialize_value_pointer_pointer(check_data, check_integer_range);
check_integer_range->minimum = minimum;
check_integer_range->maximum = maximum;
_expect_check(function, parameter, file, line, check_function,
    check_data.value, &check_integer_range->event, count);

}
```

```
/* Add an event to determine whether a parameter is within a range. */

void _expect_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType minimum, const LargestIntegralType maximum,
    const int count) {

expect_range(function, parameter, file, line, minimum, maximum,
    check_in_range, count);

}
```

```
/* Add an event to determine whether a parameter is not within a range. */

void _expect_not_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType minimum, const LargestIntegralType maximum,
```

```
    const int count) {  
  
    expect_range(function, parameter, file, line, minimum, maximum,  
  
        check_not_in_range, count);  
  
}
```

```
/* CheckParameterValue callback to check whether a value is equal to an  
* expected value. */  
  
static int check_value(const LargestIntegralType value,  
  
    const LargestIntegralType check_value_data) {  
  
    return values_equal_display_error(value, check_value_data);  
  
}
```

```
/* Add an event to check a parameter equals an expected value. */  
  
void _expect_value(  
  
    const char* const function, const char* const parameter,  
  
    const char* const file, const int line,  
  
    const LargestIntegralType value, const int count) {  
  
    _expect_check(function, parameter, file, line, check_value, value, NULL,  
  
        count);  
  
}
```

```
/* CheckParameterValue callback to check whether a value is not equal to an  
* expected value. */  
  
static int check_not_value(const LargestIntegralType value,  
  
    const LargestIntegralType check_value_data) {  
  
    return values_not_equal_display_error(value, check_value_data);
```

```
}
```

```
/* Add an event to check a parameter is not equal to an expected value. */
```

```
void _expect_not_value(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType value, const int count) {
    _expect_check(function, parameter, file, line, check_not_value, value,
        NULL, count);
}
```

```
/* CheckParameterValue callback to check whether a parameter equals a string. */
```

```
static int check_string(const LargestIntegralType value,
    const LargestIntegralType check_value_data) {
    return string_equal_display_error(
        cast_largest_integral_type_to_pointer(char*, value),
        cast_largest_integral_type_to_pointer(char*, check_value_data));
}
```

```
/* Add an event to check whether a parameter is equal to a string. */
```

```
void _expect_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count) {
    declare_initialize_value_pointer_pointer(string_pointer,
        discard_const(string));
```

```

_expect_check(function, parameter, file, line, check_string,
    string_pointer.value, NULL, count);

}

/* CheckParameterValue callback to check whether a parameter is not equals to
 * a string. */

static int check_not_string(const LargestIntegralType value,
                           const LargestIntegralType check_value_data) {
    return string_not_equal_display_error(
        cast_largest_integral_type_to_pointer(char*, value),
        cast_largest_integral_type_to_pointer(char*, check_value_data));
}

/* Add an event to check whether a parameter is not equal to a string. */

void _expect_not_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count) {
    declare_initialize_value_pointer_pointer(string_pointer,
                                              discard_const(string));
    _expect_check(function, parameter, file, line, check_not_string,
                  string_pointer.value, NULL, count);
}

/* CheckParameterValue callback to check whether a parameter equals an area of
 * memory. */

static int check_memory(const LargestIntegralType value,

```

```

    const LargestIntegralType check_value_data) {

CheckMemoryData * const check = cast_largest_integral_type_to_pointer(
    CheckMemoryData*, check_value_data);

assert_non_null(check);

return memory_equal_display_error(
    cast_largest_integral_type_to_pointer(const char*, value),
    (const char*)check->memory, check->size);

}

```

```

/* Create the callback data for check_memory() or check_not_memory() and
* register a check event. */

static void expect_memory_setup(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const void * const memory, const size_t size,
    const CheckParameterValue check_function, const int count) {

CheckMemoryData * const check_data =
    (CheckMemoryData*)malloc(sizeof(*check_data) + size);

void * const mem = (void*)(check_data + 1);

declare_initialize_value_pointer_pointer(check_data_pointer, check_data);
assert_non_null(memory);
assert_true(size);
memcpy(mem, memory, size);
check_data->memory = mem;
check_data->size = size;
_expect_check(function, parameter, file, line, check_function,
    check_data_pointer.value, &check_data->event, count);

}

```

```
/* Add an event to check whether a parameter matches an area of memory. */

void _expect_memory(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const void* const memory,
    const size_t size, const int count) {
    expect_memory_setup(function, parameter, file, line, memory, size,
        check_memory, count);
}
```

```
/* CheckParameterValue callback to check whether a parameter is not equal to
 * an area of memory. */

static int check_not_memory(const LargestIntegralType value,
    const LargestIntegralType check_value_data) {
    CheckMemoryData * const check = cast_largest_integral_type_to_pointer(
        CheckMemoryData*, check_value_data);
    assert_non_null(check);
    return memory_not_equal_display_error(
        cast_largest_integral_type_to_pointer(const char*, value),
        (const char*)(check->memory,
        check->size));
}
```

```
/* Add an event to check whether a parameter doesn't match an area of memory. */

void _expect_not_memory(
    const char* const function, const char* const parameter,
```

```
    const char* const file, const int line, const void* const memory,
    const size_t size, const int count) {

expect_memory_setup(function, parameter, file, line, memory, size,
                    check_not_memory, count);

}
```

```
/* CheckParameterValue callback that always returns 1. */

static int check_any(const LargestIntegralType value,
                     const LargestIntegralType check_value_data) {

(void)value;
(void)check_value_data;

return 1;
}
```

```
/* Add an event to allow any value for a parameter. */

void _expect_any(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const int count) {

_expect_check(function, parameter, file, line, check_any, 0, NULL,
              count);
}
```

```
void _check_expected(
    const char * const function_name, const char * const parameter_name,
    const char* file, const int line, const LargestIntegralType value) {

void *result;
```

```

const char* symbols[] = {function_name, parameter_name};

const int rc = get_symbol_value(&global_function_parameter_map_head,
                               symbols, 2, &result);

if (rc) {
    CheckParameterEvent * const check = (CheckParameterEvent*)result;
    int check_succeeded;
    global_last_parameter_location = check->location;
    check_succeeded = check->check_value(value, check->check_value_data);
    if (rc == 1) {
        free(check);
    }
    if (!check_succeeded) {
        cm_print_error(SOURCE_LOCATION_FORMAT
                      ": error: Check of parameter %s, function %s failed\n"
                      SOURCE_LOCATION_FORMAT
                      ": note: Expected parameter declared here\n",
                      file, line,
                      parameter_name, function_name,
                      global_last_parameter_location.file,
                      global_last_parameter_location.line);
        _fail(file, line);
    }
} else {
    cm_print_error(SOURCE_LOCATION_FORMAT ": error: Could not get value "
                  "to check parameter %s of function %s\n",
                  file, line,
                  parameter_name, function_name);
    if (source_location_is_set(&global_last_parameter_location)) {
        cm_print_error(SOURCE_LOCATION_FORMAT
                      ": note: Previously declared parameter value was declared here\n",

```

```
    global_last_parameter_location.file,
    global_last_parameter_location.line);
} else {
    cm_print_error("There were no previously declared parameter values "
        "for this test.\n");
}
exit_test(1);
}
```

```
/* Replacement for assert. */
void mock_assert(const int result, const char* const expression,
    const char* const file, const int line) {
    if (!result) {
        if (global_expecting_assert) {
            global_last_failed_assert = expression;
            longjmp(global_expect_assert_env, result);
        } else {
            cm_print_error("ASSERT: %s\n", expression);
            _fail(file, line);
        }
    }
}
```

```
void _assert_true(const LargestIntegralType result,
    const char * const expression,
    const char * const file, const int line) {
```

```
if (!result) {
    cm_print_error("%s\n", expression);
    _fail(file, line);
}

void _assert_return_code(const LargestIntegralType result,
    size_t rlen,
    const LargestIntegralType error,
    const char * const expression,
    const char * const file,
    const int line)
{
    LargestIntegralType valmax;

    switch (rlen) {
        case 1:
            valmax = 255;
            break;
        case 2:
            valmax = 32767;
            break;
        case 4:
            valmax = 2147483647;
            break;
        case 8:
        default:
            if (rlen > sizeof(valmax)) {
```

```

    valmax = 2147483647;
} else {
    valmax = 9223372036854775807L;
}
break;
}

if (result > valmax - 1) {
    if (error > 0) {
        cm_print_error("%s < 0, errno(\""
                      LargestIntegralTypePrintfFormatDecimal "): %s\n",
                      expression, error, strerror((int)error));
    } else {
        cm_print_error("%s < 0\n", expression);
    }
    _fail(file, line);
}

void _assert_int_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line) {
if (!values_equal_display_error(a, b)) {
    _fail(file, line);
}
}

void _assert_int_not_equal(

```

```
const LargestIntegralType a, const LargestIntegralType b,  
const char * const file, const int line) {  
  
if (!values_not_equal_display_error(a, b)) {  
  
    _fail(file, line);  
  
}  
  
}
```

```
void _assert_string_equal(const char * const a, const char * const b,  
                        const char * const file, const int line) {  
  
if (!string_equal_display_error(a, b)) {  
  
    _fail(file, line);  
  
}  
  
}
```

```
void _assert_string_not_equal(const char * const a, const char * const b,  
                            const char *file, const int line) {  
  
if (!string_not_equal_display_error(a, b)) {  
  
    _fail(file, line);  
  
}  
  
}
```

```
void _assert_memory_equal(const void * const a, const void * const b,  
                         const size_t size, const char* const file,  
                         const int line) {  
  
if (!memory_equal_display_error((const char*)a, (const char*)b, size)) {  
  
    _fail(file, line);
```

```
}

}

void _assert_memory_not_equal(const void * const a, const void * const b,
                             const size_t size, const char* const file,
                             const int line) {
    if (!memory_not_equal_display_error((const char*)a, (const char*)b,
                                         size)) {
        _fail(file, line);
    }
}

void _assert_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file,
    const int line) {
    if (!integer_in_range_display_error(value, minimum, maximum)) {
        _fail(file, line);
    }
}

void _assert_not_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file,
    const int line) {
    if (!integer_not_in_range_display_error(value, minimum, maximum)) {
        _fail(file, line);
    }
}
```

```
    }  
}  
  
}
```

```
void _assert_in_set(const LargestIntegralType value,  
                    const LargestIntegralType values[],  
                    const size_t number_of_values, const char* const file,  
                    const int line) {  
  
    CheckIntegerSet check_integer_set;  
    check_integer_set.set = values;  
    check_integer_set.size_of_set = number_of_values;  
    if (!value_in_set_display_error(value, &check_integer_set, 0)) {  
        _fail(file, line);  
    }  
}
```

```
void _assert_not_in_set(const LargestIntegralType value,  
                       const LargestIntegralType values[],  
                       const size_t number_of_values, const char* const file,  
                       const int line) {  
  
    CheckIntegerSet check_integer_set;  
    check_integer_set.set = values;  
    check_integer_set.size_of_set = number_of_values;  
    if (!value_in_set_display_error(value, &check_integer_set, 1)) {  
        _fail(file, line);  
    }  
}
```

```
/* Get the list of allocated blocks. */
```

```
static ListNode* get_allocated_blocks_list() {
    /* If it initialized, initialize the list of allocated blocks. */
    if (!global_allocated_blocks.value) {
        list_initialize(&global_allocated_blocks);
        global_allocated_blocks.value = (void*)1;
    }
    return &global_allocated_blocks;
}
```

```
static void *libc_malloc(size_t size)
```

```
{
#define malloc test_malloc
    return malloc(size);
}
```

```
static void libc_free(void *ptr)
```

```
{
#define free test_free
    free(ptr);
}
```

```
static void *libc_realloc(void *ptr, size_t size)
```

```
{
#define realloc test_realloc
    return realloc(ptr, size);
}
```

```
static void vcm_print_error(const char* const format,
                           va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);

/* It's important to use the libc malloc and free here otherwise
 * the automatic free of leaked blocks can reap the error messages
 */

static void vcm_print_error(const char* const format, va_list args)
{
    char buffer[1024];
    size_t msg_len = 0;
    va_list ap;
    int len;
    va_copy(ap, args);

    len = vsnprintf(buffer, sizeof(buffer), format, args);
    if (len < 0) {
        /* TODO */
        goto end;
    }

    if (cm_error_message == NULL) {
        /* CREATE MESSAGE */

        cm_error_message = libc_malloc(len + 1);
        if (cm_error_message == NULL) {
            /* TODO */
            goto end;
        }
    }
}
```

```
    } else {

        /* APPEND MESSAGE */

        char *tmp;

        msg_len = strlen(cm_error_message);

        tmp = libc_realloc(cm_error_message, msg_len + len + 1);

        if (tmp == NULL) {

            goto end;

        }

        cm_error_message = tmp;

    }

    if (((size_t)len) < sizeof(buffer)) {

        /* Use len + 1 to also copy '\0' */

        memcpy(cm_error_message + msg_len, buffer, len + 1);

    } else {

        vsnprintf(cm_error_message + msg_len, len, format, ap);

    }

end:

va_end(ap);

}

static void vcm_free_error(char *err_msg)

{

    libc_free(err_msg);

}

/* Use the real malloc in this function. */
```

```

#define malloc _test_malloc

void* _test_malloc(const size_t size, const char* file, const int line) {
    char* ptr;
    MallocBlockInfo *block_info;
    ListNode * const block_list = get_allocated_blocks_list();
    const size_t allocate_size = size + (MALLOC_GUARD_SIZE * 2) +
        sizeof(*block_info) + MALLOC_ALIGNMENT;
    char* const block = (char*)malloc(allocate_size);
    assert_non_null(block);

    /* Calculate the returned address. */
    ptr = (char*)((size_t)block + MALLOC_GUARD_SIZE + sizeof(*block_info) +
        MALLOC_ALIGNMENT) & ~(MALLOC_ALIGNMENT - 1);

    /* Initialize the guard blocks. */
    memset(ptr - MALLOC_GUARD_SIZE, MALLOC_GUARD_PATTERN, MALLOC_GUARD_SIZE);
    memset(ptr + size, MALLOC_GUARD_PATTERN, MALLOC_GUARD_SIZE);
    memset(ptr, MALLOC_ALLOC_PATTERN, size);

    block_info = (MallocBlockInfo*)(ptr - (MALLOC_GUARD_SIZE +
        sizeof(*block_info)));
    set_source_location(&block_info->location, file, line);
    block_info->allocated_size = allocate_size;
    block_info->size = size;
    block_info->block = block;
    block_info->node.value = block_info;
    list_add(block_list, &block_info->node);
    return ptr;
}

```

```
#define malloc test_malloc

void* _test_calloc(const size_t number_of_elements, const size_t size,
                   const char* file, const int line) {
    void* const ptr = _test_malloc(number_of_elements * size, file, line);
    if (ptr) {
        memset(ptr, 0, number_of_elements * size);
    }
    return ptr;
}

/* Use the real free in this function. */
#undef free

void _test_free(void* const ptr, const char* file, const int line) {
    unsigned int i;
    char *block = discard_const_p(char, ptr);
    MallocBlockInfo *block_info;

    if (ptr == NULL) {
        return;
    }

    _assert_true(cast_ptr_to_largest_integral_type(ptr), "ptr", file, line);
    block_info = (MallocBlockInfo*)(block - (MALLOC_GUARD_SIZE +
                                              sizeof(*block_info)));
    /* Check the guard blocks. */
    {

```

```

char *guards[2] = {block - MALLOC_GUARD_SIZE,
                  block + block_info->size};

for (i = 0; i < ARRAY_SIZE(guards); i++) {
    unsigned int j;

    char * const guard = guards[i];
    for (j = 0; j < MALLOC_GUARD_SIZE; j++) {
        const char diff = guard[j] - MALLOC_GUARD_PATTERN;
        if (diff) {
            cm_print_error(SOURCE_LOCATION_FORMAT
                           ": error: Guard block of %p size=%lu is corrupt\n"
                           SOURCE_LOCATION_FORMAT ": note: allocated here at %p\n",
                           file, line,
                           ptr, (unsigned long)block_info->size,
                           block_info->location.file, block_info->location.line,
                           (void *)&guard[j]);
            _fail(file, line);
        }
    }
}

list_remove(&block_info->node, NULL, NULL);

block = discard_const_p(char, block_info->block);
memset(block, MALLOC_FREE_PATTERN, block_info->allocated_size);
free(block);
}

#define free test_free

#endif

```

```
void *_test_realloc(void *ptr,
                    const size_t size,
                    const char *file,
                    const int line)

{
    MallocBlockInfo *block_info;
    char *block = ptr;
    size_t block_size = size;
    void *new;

    if (ptr == NULL) {
        return _test_malloc(size, file, line);
    }

    if (size == 0) {
        _test_free(ptr, file, line);
        return NULL;
    }

    block_info = (MallocBlockInfo*)(block - (MALLOC_GUARD_SIZE +
                                              sizeof(*block_info)));
}

new = _test_malloc(size, file, line);
if (new == NULL) {
    return NULL;
}

if (block_info->size < size) {
    block_size = block_info->size;
```

```
}

memcpy(new, ptr, block_size);

/* Free previous memory */
_test_free(ptr, file, line);

return new;
}

#define realloc test_realloc

/* Crudely checkpoint the current heap state. */

static const ListNode* check_point_allocated_blocks() {

    return get_allocated_blocks_list()->prev;
}

/* Display the blocks allocated after the specified check point. This
 * function returns the number of blocks displayed. */

static int display_allocated_blocks(const ListNode * const check_point) {

    const ListNode * const head = get_allocated_blocks_list();

    const ListNode *node;

    int allocated_blocks = 0;

    assert_non_null(check_point);

    assert_non_null(check_point->next);

    for (node = check_point->next; node != head; node = node->next) {

        const MallocBlockInfo * const block_info =
            (const MallocBlockInfo*)node->value;
```

```

assert_non_null(block_info);

if (!allocated_blocks) {
    cm_print_error("Blocks allocated...\n");
}

cm_print_error(SOURCE_LOCATION_FORMAT ": note: block %p allocated here\n",
               block_info->location.file,
               block_info->location.line,
               block_info->block);

allocated_blocks++;
}

return allocated_blocks;
}

```

```

/* Free all blocks allocated after the specified check point. */

static void free_allocated_blocks(const ListNode * const check_point) {
    const ListNode * const head = get_allocated_blocks_list();
    const ListNode *node;
    assert_non_null(check_point);

    node = check_point->next;
    assert_non_null(node);

    while (node != head) {
        MallocBlockInfo * const block_info = (MallocBlockInfo*)node->value;
        node = node->next;
        free(discard_const_p(char, block_info) + sizeof(*block_info) + MALLOC_GUARD_SIZE);
    }
}

```

```
}

/* Fail if any any blocks are allocated after the specified check point. */

static void fail_if_blocks_allocated(const ListNode * const check_point,
                                     const char * const test_name) {
    const int allocated_blocks = display_allocated_blocks(check_point);
    if (allocated_blocks) {
        free_allocated_blocks(check_point);
        cm_print_error("ERROR: %s leaked %d block(s)\n", test_name,
                      allocated_blocks);
        exit_test(1);
    }
}

void _fail(const char * const file, const int line) {
    enum cm_message_output output = cm_get_output();

    switch(output) {
        case CM_OUTPUT_STDOUT:
            cm_print_error("[ LINE ] --- " SOURCE_LOCATION_FORMAT ": error: Failure!", file, line);
            break;
        default:
            cm_print_error(SOURCE_LOCATION_FORMAT ": error: Failure!", file, line);
            break;
    }
    exit_test(1);
}
```

```
#ifndef _WIN32

static void exception_handler(int sig) {
    const char *sig_strerror = "";

#ifndef HAVE_STRSIGNAL
    sig_strerror = strsignal(sig);
#endif

    cm_print_error("Test failed with exception: %s(%d)",
                  sig_strerror, sig);
    exit_test(1);
}

#else /* _WIN32 */

static LONG WINAPI exception_filter(EXCEPTION_POINTERS *exception_pointers) {
    EXCEPTION_RECORD * const exception_record =
        exception_pointers->ExceptionRecord;
    const DWORD code = exception_record->ExceptionCode;
    unsigned int i;

    for (i = 0; i < ARRAY_SIZE(exception_codes); i++) {
        const ExceptionCodeInfo * const code_info = &exception_codes[i];
        if (code == code_info->code) {
            static int shown_debug_message = 0;
            fflush(stdout);
            cm_print_error("%s occurred at %p.\n", code_info->description,
                           exception_record->ExceptionAddress);
        }
    }
}
```

```

if (!shown_debug_message) {
    cm_print_error(
        "\n"
        "To debug in Visual Studio...\n"
        "1. Select menu item File->Open Project\n"
        "2. Change 'Files of type' to 'Executable Files'\n"
        "3. Open this executable.\n"
        "4. Select menu item Debug->Start\n"
        "\n"
        "Alternatively, set the environment variable \n"
        "UNIT_TESTING_DEBUG to 1 and rebuild this executable, \n"
        "then click 'Debug' in the popup dialog box.\n"
        "\n");
    shown_debug_message = 1;
}
exit_test(0);
return EXCEPTION_EXECUTE_HANDLER;
}
}

return EXCEPTION_CONTINUE_SEARCH;
}

#endif /* !_WIN32 */

```

```

void cm_print_error(const char * const format, ...)
{
    va_list args;
    va_start(args, format);
    if (cm_error_message_enabled) {
        vcm_print_error(format, args);
    }
}

```

```
    } else {
        vprint_error(format, args);
    }

    va_end(args);
}

/* Standard output and error print methods. */

void vprint_message(const char* const format, va_list args) {
    char buffer[1024];

    vsnprintf(buffer, sizeof(buffer), format, args);

    printf("%s", buffer);
    fflush(stdout);

#ifndef _WIN32
    OutputDebugString(buffer);
#endif /* _WIN32 */

}

void vprint_error(const char* const format, va_list args) {
    char buffer[1024];

    vsnprintf(buffer, sizeof(buffer), format, args);

    fprintf(stderr, "%s", buffer);
    fflush(stderr);

#ifndef _WIN32
    OutputDebugString(buffer);
#endif /* _WIN32 */

}
```

```
void print_message(const char* const format, ...) {
    va_list args;
    va_start(args, format);
    vprint_message(format, args);
    va_end(args);
}

void print_error(const char* const format, ...) {
    va_list args;
    va_start(args, format);
    vprint_error(format, args);
    va_end(args);
}

/* New formatter */

static enum cm_message_output cm_get_output(void)
{
    enum cm_message_output output = global_msg_output;
    char *env;

    env = getenv("CMOCKA_MESSAGE_OUTPUT");
    if (env != NULL) {
        if (strcasecmp(env, "STDOOUT") == 0) {
            output = CM_OUTPUT_STDOOUT;
        } else if (strcasecmp(env, "SUBUNIT") == 0) {
            output = CM_OUTPUT_SUBUNIT;
        } else if (strcasecmp(env, "TAP") == 0) {
            output = CM_OUTPUT_TAP;
        }
    }
}
```

```
    } else if (strcasecmp(env, "XML") == 0) {

        output = CM_OUTPUT_XML;

    }

}

return output;
}

enum cm_printf_type {

    PRINTF_TEST_START,
    PRINTF_TEST_SUCCESS,
    PRINTF_TEST_FAILURE,
    PRINTF_TEST_ERROR,
    PRINTF_TEST_SKIPPED,
};

static int xml_printed;
static int file_append;

static void cmprintf_group_finish_xml(const char *group_name,
                                      size_t total_executed,
                                      size_t total_failed,
                                      size_t total_errors,
                                      size_t total_skipped,
                                      double total_runtime,
                                      struct CMUnitTestState *cm_tests)

{
    FILE *fp = stdout;
    int file_opened = 0;
```

```
int multiple_files = 0;
char *env;
size_t i;

env = getenv("CMOCKA_XML_FILE");
if (env != NULL) {
    char buf[1024];
    int rc;

    snprintf(buf, sizeof(buf), "%s", env);

    rc = c_strreplace(buf, sizeof(buf), "%g", group_name, &multiple_files);
    if (rc < 0) {
        snprintf(buf, sizeof(buf), "%s", env);
    }
}

fp = fopen(buf, "r");
if (fp == NULL) {
    fp = fopen(buf, "w");
    if (fp != NULL) {
        file_append = 1;
        file_opened = 1;
    } else {
        fp = stderr;
    }
} else {
    fclose(fp);
    if (file_append) {
        fp = fopen(buf, "a");
    }
}
```

```

if (fp != NULL) {
    file_opened = 1;
    xml_printed = 1;
} else {
    fp = stderr;
}
} else {
    fp = stderr;
}
}

if (!xml_printed || (file_opened && !file_append)) {
    fprintf(fp, "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n");
    if (!file_opened) {
        xml_printed = 1;
    }
}

fprintf(fp, "<testsuites>\n");
fprintf(fp, " <testsuite name=\"%s\" time=\"%3f\" "
    "tests=\"%u\" failures=\"%u\" errors=\"%u\" skipped=\"%u\" >\n",
    group_name,
    total_runtime * 1000, /* miliseconds */
    (unsigned)total_executed,
    (unsigned)total_failed,
    (unsigned)total_errors,
    (unsigned)total_skipped);

```

```
for (i = 0; i < total_executed; i++) {  
    struct CMUnitTestState *cmtest = &cm_tests[i];  
  
    fprintf(fp, "  < testcase name=\"%s\" time=\"% .3f\" >\n",  
            cmtest->test->name, cmtest->runtime * 1000);  
  
    switch (cmtest->status) {  
        case CM_TEST_ERROR:  
        case CM_TEST_FAILED:  
            if (cmtest->error_message != NULL) {  
                fprintf(fp, "    < failure ><![CDATA[%s]]></ failure >\n",  
                        cmtest->error_message);  
            } else {  
                fprintf(fp, "    < failure message=\"Unknown error\" />\n");  
            }  
            break;  
        case CM_TEST_SKIPPED:  
            fprintf(fp, "    < skipped >\n");  
            break;  
  
        case CM_TEST_PASSED:  
        case CM_TEST_NOT_STARTED:  
            break;  
    }  
  
    fprintf(fp, "  </ testcase >\n");  
}  
  
fprintf(fp, " </ testsuite >\n");
```

```

fprintf(fp, "</testsuites>\n");

if (file_opened) {
    fclose(fp);
}

}

static void cmprintf_group_start_standard(const size_t num_tests)
{
    print_message("[=====] Running %u test(s).\n",
        (unsigned)num_tests);
}

static void cmprintf_group_finish_standard(size_t total_executed,
                                           size_t total_passed,
                                           size_t total_failed,
                                           size_t total_errors,
                                           size_t total_skipped,
                                           struct CMUnitTestState *cm_tests)
{
    size_t i;

    print_message("[=====] %u test(s) run.\n", (unsigned)total_executed);
    print_error("[ PASSED ] %u test(s).\n",
        (unsigned)(total_passed));

    if (total_skipped) {
        print_error("[ SKIPPED ] %"PRIuS " test(s), listed below:\n", total_skipped);
        for (i = 0; i < total_executed; i++) {

```

```

    struct CMUnitTestState *cmtest = &cm_tests[i];

    if (cmtest->status == CM_TEST_SKIPPED) {
        print_error("[ SKIPPED ] %s\n", cmtest->test->name);
    }
}

print_error("\n %u SKIPPED TEST(S)\n", (unsigned)(total_skipped));
}

if (total_failed) {
    print_error("[ FAILED ] %"PRIuS " test(s), listed below:\n", total_failed);
    for (i = 0; i < total_executed; i++) {
        struct CMUnitTestState *cmtest = &cm_tests[i];

        if (cmtest->status == CM_TEST_FAILED) {
            print_error("[ FAILED ] %s\n", cmtest->test->name);
        }
    }
    print_error("\n %u FAILED TEST(S)\n",
               (unsigned)(total_failed + total_errors));
}
}

static void cmprintf_standard(enum cm_printf_type type,
                             const char *test_name,
                             const char *error_message)
{
    switch (type) {
        case PRINTF_TEST_START:

```

```
    print_message("[ RUN    ] %s\n", test_name);
    break;

case PRINTF_TEST_SUCCESS:
    print_message("[    OK ] %s\n", test_name);
    break;

case PRINTF_TEST_FAILURE:
    if (error_message != NULL) {
        print_error("[ ERROR  ] --- %s\n", error_message);
    }
    print_message("[ FAILED ] %s\n", test_name);
    break;

case PRINTF_TEST_SKIPPED:
    print_message("[ SKIPPED ] %s\n", test_name);
    break;

case PRINTF_TEST_ERROR:
    if (error_message != NULL) {
        print_error("%s\n", error_message);
    }
    print_error("[ ERROR  ] %s\n", test_name);
    break;
}

}

static void cmprintf_group_start_tap(const size_t num_tests)
{
    print_message("\t1..%u\n", (unsigned)num_tests);
}

static void cmprintf_group_finish_tap(const char *group_name,
```

```

        size_t total_executed,
        size_t total_passed,
        size_t total_skipped)

{

    const char *status = "not ok";

    if (total_passed + total_skipped == total_executed) {

        status = "ok";
    }

    print_message("%s - %s\n", status, group_name);
}

static void cmprintf_tap(enum cm_printf_type type,
                       uint32_t test_number,
                       const char *test_name,
                       const char *error_message)

{
    switch (type) {

        case PRINTF_TEST_START:
            break;

        case PRINTF_TEST_SUCCESS:
            print_message("\tok %u - %s\n", (unsigned)test_number, test_name);
            break;

        case PRINTF_TEST_FAILURE:
            print_message("\tnot ok %u - %s\n", (unsigned)test_number, test_name);
            if (error_message != NULL) {

                char *msg;
                char *p;

                msg = strdup(error_message);

```

```
if (msg == NULL) {
    return;
}

p = msg;

while (p[0] != '\0') {
    char *q = p;

    p = strchr(q, '\n');

    if (p != NULL) {
        p[0] = '\0';

    }

    print_message("\t# %s\n", q);

    if (p == NULL) {
        break;
    }
    p++;
}

libc_free(msg);
}

break;

case PRINTF_TEST_SKIPPED:
    print_message("\tnot ok %u # SKIP %s\n", (unsigned)test_number, test_name);
    break;

case PRINTF_TEST_ERROR:
    print_message("\tnot ok %u - %s %s\n",
        (unsigned)test_number, test_name, error_message);
```

```
        break;
    }
}

static void cmprintf_subunit(enum cm_printf_type type,
                           const char *test_name,
                           const char *error_message)
{
    switch (type) {
    case PRINTF_TEST_START:
        print_message("test: %s\n", test_name);
        break;
    case PRINTF_TEST_SUCCESS:
        print_message("success: %s\n", test_name);
        break;
    case PRINTF_TEST_FAILURE:
        print_message("failure: %s", test_name);
        if (error_message != NULL) {
            print_message(" [\\n%s]\\n", error_message);
        }
        break;
    case PRINTF_TEST_SKIPPED:
        print_message("skip: %s\\n", test_name);
        break;
    case PRINTF_TEST_ERROR:
        print_message("error: %s [ %s ]\\n", test_name, error_message);
        break;
    }
}
```

```
static void cmprintf_group_start(const size_t num_tests)

{
    enum cm_message_output output;

    output = cm_get_output();

    switch (output) {
        case CM_OUTPUT_STDOUT:
            cmprintf_group_start_standard(num_tests);
            break;
        case CM_OUTPUT_SUBUNIT:
            break;
        case CM_OUTPUT_TAP:
            cmprintf_group_start_tap(num_tests);
            break;
        case CM_OUTPUT_XML:
            break;
    }
}
```

```
{  
    enum cm_message_output output;  
  
    output = cm_get_output();  
  
    switch (output) {  
        case CM_OUTPUT_STDOUT:  
            cmprintf_group_finish_standard(total_executed,  
                total_passed,  
                total_failed,  
                total_errors,  
                total_skipped,  
                cm_tests);  
            break;  
        case CM_OUTPUT_SUBUNIT:  
            break;  
        case CM_OUTPUT_TAP:  
            cmprintf_group_finish_tap(group_name, total_executed, total_passed, total_skipped);  
            break;  
        case CM_OUTPUT_XML:  
            cmprintf_group_finish_xml(group_name,  
                total_executed,  
                total_failed,  
                total_errors,  
                total_skipped,  
                total_runtime,  
                cm_tests);  
            break;  
    }  
}
```

```
}

static void cmprintf(enum cm_printf_type type,
                    size_t test_number,
                    const char *test_name,
                    const char *error_message)
{
    enum cm_message_output output;

    output = cm_get_output();

    switch (output) {
        case CM_OUTPUT_STDOUT:
            cmprintf_standard(type, test_name, error_message);
            break;
        case CM_OUTPUT_SUBUNIT:
            cmprintf_subunit(type, test_name, error_message);
            break;
        case CM_OUTPUT_TAP:
            cmprintf_tap(type, test_number, test_name, error_message);
            break;
        case CM_OUTPUT_XML:
            break;
    }
}

void cmocka_set_message_output(enum cm_message_output output)
{
    global_msg_output = output;
```

```
}
```

```
*****
```

```
* TIME CALCULATIONS
```

```
*****
```

```
#ifdef HAVE_STRUCT_TIMESPEC
```

```
static struct timespec cm_tspeccdiff(struct timespec time1,  
                                     struct timespec time0)
```

```
{
```

```
    struct timespec ret;
```

```
    int xsec = 0;
```

```
    int sign = 1;
```

```
    if (time0.tv_nsec > time1.tv_nsec) {
```

```
        xsec = (int) ((time0.tv_nsec - time1.tv_nsec) / (1E9 + 1));
```

```
        time0.tv_nsec -= (long int) (1E9 * xsec);
```

```
        time0.tv_sec += xsec;
```

```
}
```

```
    if ((time1.tv_nsec - time0.tv_nsec) > 1E9) {
```

```
        xsec = (int) ((time1.tv_nsec - time0.tv_nsec) / 1E9);
```

```
        time0.tv_nsec += (long int) (1E9 * xsec);
```

```
        time0.tv_sec -= xsec;
```

```
}
```

```
    ret.tv_sec = time1.tv_sec - time0.tv_sec;
```

```
    ret.tv_nsec = time1.tv_nsec - time0.tv_nsec;
```



```

        CMFixtureFunction teardown_func,
        void ** const volatile state,
        const void *const heap_check_point)

{

    const ListNode * const volatile check_point = (const ListNode*)
        (heap_check_point != NULL ?
         heap_check_point : check_point_allocated_blocks());

    int handle_exceptions = 1;
    void *current_state = NULL;
    int rc = 0;

    /* FIXME check only one test or fixture is set */

    /* Detect if we should handle exceptions */
    #ifdef _WIN32
        handle_exceptions = !IsDebuggerPresent();
    #endif /* _WIN32 */
    #ifdef UNIT_TESTING_DEBUG
        handle_exceptions = 0;
    #endif /* UNIT_TESTING_DEBUG */

    if (handle_exceptions) {
        #ifndef _WIN32
            unsigned int i;
            for (i = 0; i < ARRAY_SIZE(exception_signals); i++) {
                default_signal_functions[i] = signal(
                    exception_signals[i], exception_handler);
            }
        #endif /* _WIN32 */
    }
}
```

```
#else /* _WIN32 */

    previous_exception_filter = SetUnhandledExceptionFilter(
        exception_filter);

#endif /* !_WIN32 */

}

/* Init the test structure */

initialize_testing(function_name);

global_running_test = 1;

if (cm_setjmp(global_run_test_env) == 0) {

    if (test_func != NULL) {

        test_func(state != NULL ? state : &current_state);

        fail_if_blocks_allocated(check_point, function_name);

        rc = 0;

    } else if (setup_func != NULL) {

        rc = setup_func(state != NULL ? state : &current_state);

        /*
         * For setup we can ignore any allocated blocks. We just need to
         * ensure they're deallocated on tear down.
         */

    } else if (teardown_func != NULL) {

        rc = teardown_func(state != NULL ? state : &current_state);

        fail_if_blocks_allocated(check_point, function_name);

    } else {


```

```

/* ERROR */
}

fail_if_leftover_values(function_name);

global_running_test = 0;

} else {

/* TEST FAILED */

global_running_test = 0;

rc = -1;

}

teardown_testing(function_name);

if (handle_exceptions) {

#ifndef _WIN32

unsigned int i;

for (i = 0; i < ARRAY_SIZE(exception_signals); i++) {

signal(exception_signals[i], default_signal_functions[i]);

}

#else /* !_WIN32 */

if (previous_exception_filter) {

SetUnhandledExceptionFilter(previous_exception_filter);

previous_exception_filter = NULL;

}

#endif /* !_WIN32 */

}

return rc;
}

```

```
static int cmocka_run_group_fixture(const char *function_name,
```

```
    CMFixtureFunction setup_func,
    CMFixtureFunction teardown_func,
    void **state,
    const void *const heap_check_point)

{

    int rc;

    if (setup_func != NULL) {

        rc = cmocka_run_one_test_or_fixture(function_name,
                                             NULL,
                                             setup_func,
                                             NULL,
                                             state,
                                             heap_check_point);

    } else {

        rc = cmocka_run_one_test_or_fixture(function_name,
                                             NULL,
                                             NULL,
                                             teardown_func,
                                             state,
                                             heap_check_point);
    }

    return rc;
}
```

```
static int cmocka_run_one_tests(struct CMUnitTestState *test_state)
{
#ifndef HAVE_STRUCT_TIMESPEC
```

```

struct timespec start = {
    .tv_sec = 0,
    .tv_nsec = 0,
};

struct timespec finish = {
    .tv_sec = 0,
    .tv_nsec = 0,
};

#endif

int rc = 0;

/* Run setup */
if (test_state->test->setup_func != NULL) {
    /* Setup the memory check point, it will be evaluated on teardown */
    test_state->check_point = check_point_allocated_blocks();

    rc = cmocka_run_one_test_or_fixture(test_state->test->name,
                                        NULL,
                                        test_state->test->setup_func,
                                        NULL,
                                        &test_state->state,
                                        test_state->check_point);

    if (rc != 0) {
        test_state->status = CM_TEST_ERROR;
        cm_print_error("Test setup failed");
    }
}

/* Run test */

```

```

#ifndef HAVE_STRUCT_TIMESPEC

CMOCKA_CLOCK_GETTIME(CLOCK_REALTIME, &start);

#endif

if (rc == 0) {

    rc = cmocka_run_one_test_or_fixture(test_state->test->name,
                                         test_state->test->test_func,
                                         NULL,
                                         NULL,
                                         &test_state->state,
                                         NULL);

    if (rc == 0) {

        test_state->status = CM_TEST_PASSED;

    } else {

        if (global_skip_test) {

            test_state->status = CM_TEST_SKIPPED;

            global_skip_test = 0; /* Do not skip the next test */

        } else {

            test_state->status = CM_TEST_FAILED;

        }

    }

    rc = 0;

}

test_state->runtime = 0.0;

```

```

#ifndef HAVE_STRUCT_TIMESPEC

CMOCKA_CLOCK_GETTIME(CLOCK_REALTIME, &finish);

test_state->runtime = cm_secdiff(finish, start);

```

```
#endif

/* Run teardown */

if (rc == 0 && test_state->test->teardown_func != NULL) {
    rc = cmocka_run_one_test_or_fixture(test_state->test->name,
                                         NULL,
                                         NULL,
                                         test_state->test->teardown_func,
                                         &test_state->state,
                                         test_state->check_point);

    if (rc != 0) {
        test_state->status = CM_TEST_ERROR;
        cm_print_error("Test teardown failed");
    }
}

test_state->error_message = cm_error_message;
cm_error_message = NULL;

return rc;
}

int _cmocka_run_group_tests(const char *group_name,
                           const struct CMUnitTest * const tests,
                           const size_t num_tests,
                           CMFixtureFunction group_setup,
                           CMFixtureFunction group_teardown)
{
    struct CMUnitTestState *cm_tests;
```

```

const ListNode *group_check_point = check_point_allocated_blocks();

void *group_state = NULL;

size_t total_tests = 0;

size_t total_failed = 0;

size_t total_passed = 0;

size_t total_executed = 0;

size_t total_errors = 0;

size_t total_skipped = 0;

double total_runtime = 0;

size_t i;

int rc;

/* Make sure LargestIntegralType is at least the size of a pointer. */
assert_true(sizeof(LargestIntegralType) >= sizeof(void*));

cm_tests = (struct CMUnitTestState *)libc_malloc(sizeof(struct CMUnitTestState) * num_tests);

if (cm_tests == NULL) {

    return -1;

}

/* Setup cmocka test array */

for (i = 0; i < num_tests; i++) {

    if (tests[i].name != NULL &&

        (tests[i].test_func != NULL

         || tests[i].setup_func != NULL

         || tests[i].teardown_func != NULL)) {

        cm_tests[i] = (struct CMUnitTestState) {

            .test = &tests[i],


            .status = CM_TEST_NOT_STARTED,

```

```

.state = NULL,
};

total_tests++;

}

}

cmprintf_group_start(total_tests);

rc = 0;

/* Run group setup */
if (group_setup != NULL) {
    rc = cmocka_run_group_fixture("cmocka_group_setup",
        group_setup,
        NULL,
        &group_state,
        group_check_point);
}

if (rc == 0) {
    /* Execute tests */
    for (i = 0; i < total_tests; i++) {
        struct CMUnitTestFixture *cmtest = &cm_tests[i];
        size_t test_number = i + 1;

        cmprintf(PRINTF_TEST_START, test_number, cmtest->test->name, NULL);

        if (group_state != NULL) {
            cmtest->state = group_state;

```

```
} else if (cmtest->test->initial_state != NULL) {  
    cmtest->state = cmtest->test->initial_state;  
}  
  
rc = cmocka_run_one_tests(cmtest);  
total_executed++;  
total_runtime += cmtest->runtime;  
if (rc == 0) {  
    switch (cmtest->status) {  
        case CM_TEST_PASSED:  
            cmprintf(PRINTF_TEST_SUCCESS,  
                     test_number,  
                     cmtest->test->name,  
                     cmtest->error_message);  
            total_passed++;  
            break;  
        case CM_TEST_SKIPPED:  
            cmprintf(PRINTF_TEST_SKIPPED,  
                     test_number,  
                     cmtest->test->name,  
                     cmtest->error_message);  
            total_skipped++;  
            break;  
        case CM_TEST_FAILED:  
            cmprintf(PRINTF_TEST_FAILURE,  
                     test_number,  
                     cmtest->test->name,  
                     cmtest->error_message);  
            total_failed++;  
    }  
}
```

```
        break;

    default:
        cmprintf(PRINTF_TEST_ERROR,
                 test_number,
                 cmtest->test->name,
                 "Internal cmocka error");

        total_errors++;

        break;
    }

} else {
    cmprintf(PRINTF_TEST_ERROR,
             test_number,
             cmtest->test->name,
             "Could not run the test - check test fixtures");

    total_errors++;

}

}

} else {
    if (cm_error_message != NULL) {
        print_error("[ ERROR ] --- %s\n", cm_error_message);
        vcm_free_error(cm_error_message);
        cm_error_message = NULL;
    }

    cmprintf(PRINTF_TEST_ERROR, 0,
             group_name, "[ FAILED ] GROUP SETUP");
    total_errors++;
}

/* Run group teardown */
```

```

if (group_teardown != NULL) {
    rc = cmocka_run_group_fixture("cmocka_group_teardown",
        NULL,
        group_teardown,
        &group_state,
        group_check_point);

    if (rc != 0) {
        if (cm_error_message != NULL) {
            print_error("[ ERROR ] --- %s\n", cm_error_message);
            vcm_free_error(cm_error_message);
            cm_error_message = NULL;
        }
        cmprintf(PRINTF_TEST_ERROR, 0,
            group_name, "[ FAILED ] GROUP TEARDOWN");
    }
}

cmprintf_group_finish(group_name,
    total_executed,
    total_passed,
    total_failed,
    total_errors,
    total_skipped,
    total_runtime,
    cm_tests);

for (i = 0; i < total_tests; i++) {
    vcm_free_error(discard_const_p(char, cm_tests[i].error_message));
}

```

```

    libc_free(cm_tests);

    fail_if_blocks_allocated(group_check_point, "cmocka_group_tests");

    return total_failed + total_errors;
}

/*****************
 * DEPRECATED TEST RUNNER
 *****************/
int _run_test(
    const char * const function_name, const UnitTestFunction Function,
    void ** const volatile state, const UnitTestFunctionType function_type,
    const void* const heap_check_point) {
    const ListNode * const volatile check_point = (const ListNode*)
        (heap_check_point ?
            heap_check_point : check_point_allocated_blocks());
    void *current_state = NULL;
    volatile int rc = 1;
    int handle_exceptions = 1;

#ifdef _WIN32
    handle_exceptions = !IsDebuggerPresent();
#endif /* _WIN32 */

#ifdef UNIT_TESTING_DEBUG
    handle_exceptions = 0;
#endif /* UNIT_TESTING_DEBUG */

    cm_error_message_enabled = 0;
}

```

```

if (handle_exceptions) {

#ifndef _WIN32

    unsigned int i;

    for (i = 0; i < ARRAY_SIZE(exception_signals); i++) {

        default_signal_functions[i] = signal(
            exception_signals[i], exception_handler);

    }

#else /* !_WIN32 */

    previous_exception_filter = SetUnhandledExceptionFilter(
        exception_filter);

#endif /* !_WIN32 */

}

if (function_type == UNIT_TEST_FUNCTION_TYPE_TEST) {

    print_message("[ RUN    ] %s\n", function_name);

}

initialize_testing(function_name);

global_running_test = 1;

if (cm_setjmp(global_run_test_env) == 0) {

    Function(state ? state : &current_state);

    fail_if_leftover_values(function_name);

/* If this is a setup function then ignore any allocated blocks
 * only ensure they're deallocated on tear down. */

    if (function_type != UNIT_TEST_FUNCTION_TYPE_SETUP) {

        fail_if_blocks_allocated(check_point, function_name);

    }

    global_running_test = 0;
}

```

```
if (function_type == UNIT_TEST_FUNCTION_TYPE_TEST) {
    print_message("[    OK ] %s\n", function_name);
}
rc = 0;
} else {
    global_running_test = 0;
    print_message("[ FAILED ] %s\n", function_name);
}
teardown_testing(function_name);

if (handle_exceptions) {
#ifndef _WIN32
    unsigned int i;
    for (i = 0; i < ARRAY_SIZE(exception_signals); i++) {
        signal(exception_signals[i], default_signal_functions[i]);
    }
#else /* _WIN32 */
    if (previous_exception_filter) {
        SetUnhandledExceptionFilter(previous_exception_filter);
        previous_exception_filter = NULL;
    }
#endif /* !_WIN32 */
}

return rc;
}
```

```

int _run_tests(const UnitTest * const tests, const size_t number_of_tests) {
    /* Whether to execute the next test. */
    int run_next_test = 1;

    /* Whether the previous test failed. */
    int previous_test_failed = 0;

    /* Whether the previous setup failed. */
    int previous_setup_failed = 0;

    /* Check point of the heap state. */
    const ListNode * const check_point = check_point_allocated_blocks();

    /* Current test being executed. */
    size_t current_test = 0;

    /* Number of tests executed. */
    size_t tests_executed = 0;

    /* Number of failed tests. */
    size_t total_failed = 0;

    /* Number of setup functions. */
    size_t setups = 0;

    /* Number of teardown functions. */
    size_t teardowns = 0;

    size_t i;

    /*
     * A stack of test states. A state is pushed on the stack
     * when a test setup occurs and popped on tear down.
     */
    TestState* test_states =
        (TestState*)malloc(number_of_tests * sizeof(*test_states));
    /* The number of test states which should be 0 at the end */
    long number_of_test_states = 0;

    /* Names of the tests that failed. */

```

```

const char** failed_names = (const char**)malloc(number_of_tests *
                                                sizeof(*failed_names));

void **current_state = NULL;

/* Count setup and teardown functions */

for (i = 0; i < number_of_tests; i++) {
    const UnitTest * const test = &tests[i];

    if (test->function_type == UNIT_TEST_FUNCTION_TYPE_SETUP) {
        setups++;
    }

    if (test->function_type == UNIT_TEST_FUNCTION_TYPE_TEARDOWN) {
        teardowns++;
    }
}

print_message("[=====] Running %"PRIIdS " test(s).\n",
             number_of_tests - setups - teardowns);

/* Make sure LargestIntegralType is at least the size of a pointer.*/
assert_true(sizeof(LargestIntegralType) >= sizeof(void*));

while (current_test < number_of_tests) {

    const ListNode *test_check_point = NULL;
    TestState *current_TestState;
    const UnitTest * const test = &tests[current_test++];

    if (!test->function) {
        continue;
    }
}

```

```
}

switch (test->function_type) {

case UNIT_TEST_FUNCTION_TYPE_TEST:

    if (! previous_setup_failed) {

        run_next_test = 1;

    }

    break;

case UNIT_TEST_FUNCTION_TYPE_SETUP: {

    /* Checkpoint the heap before the setup. */

    current_TestState = &test_states[number_of_test_states++];

    current_TestState->check_point = check_point_allocated_blocks();

    test_check_point = current_TestState->check_point;

    current_state = &current_TestState->state;

    *current_state = NULL;

    run_next_test = 1;

    break;

}

case UNIT_TEST_FUNCTION_TYPE_TEARDOWN:

    /* Check the heap based on the last setup checkpoint. */

    assert_true(number_of_test_states);

    current_TestState = &test_states[--number_of_test_states];

    test_check_point = current_TestState->check_point;

    current_state = &current_TestState->state;

    break;

default:

    print_error("Invalid unit test function type %d\n",
               test->function_type);

    exit_test(1);
```

```
        break;

    }

if (run_next_test) {

    int failed = _run_test(test->name, test->function, current_state,
                          test->function_type, test_check_point);

    if (failed) {
        failed_names[total_failed] = test->name;
    }

switch (test->function_type) {

case UNIT_TEST_FUNCTION_TYPE_TEST:

    previous_test_failed = failed;
    total_failed += failed;
    tests_executed++;
    break;

case UNIT_TEST_FUNCTION_TYPE_SETUP:

    if (failed) {
        total_failed++;
        tests_executed++;
        /* Skip forward until the next test or setup function. */
        run_next_test = 0;
        previous_setup_failed = 1;
    }
    previous_test_failed = 0;
    break;

case UNIT_TEST_FUNCTION_TYPE_TEARDOWN:
```

```

/* If this test failed. */
if (failed && !previous_test_failed) {

    total_failed++;

}

break;

default:

#ifndef _HPUX

    assert_null("BUG: shouldn't be here!");

#endif

break;

}

}

}

print_message("[=====] %"PRIuS " test(s) run.\n", tests_executed);

print_error("[ PASSED ] %"PRIuS " test(s).\n", tests_executed - total_failed);

if (total_failed > 0) {

    print_error("[ FAILED ] %"PRIuS " test(s), listed below:\n", total_failed);

    for (i = 0; i < total_failed; i++) {

        print_error("[ FAILED ] %s\n", failed_names[i]);

    }

} else {

    print_error("\n %"PRIuS " FAILED TEST(S)\n", total_failed);

}

if (number_of_test_states != 0) {

    print_error("[ ERROR ] Mismatched number of setup %"PRIuS " and "

        "teardown %"PRIuS " functions\n", setups, teardowns);
}

```

```

    total_failed = (size_t)-1;

}

free(test_states);
free((void*)failed_names);

fail_if_blocks_allocated(check_point, "run_tests");

return (int)total_failed;
}

int _run_group_tests(const UnitTest * const tests, const size_t number_of_tests)
{
    UnitTestFunction setup = NULL;
    const char *setup_name;
    size_t num_setups = 0;
    UnitTestFunction teardown = NULL;
    const char *teardown_name;
    size_t num_teardowns = 0;
    size_t current_test = 0;
    size_t i;

    /* Number of tests executed. */
    size_t tests_executed = 0;
    /* Number of failed tests. */
    size_t total_failed = 0;
    /* Check point of the heap state. */
    const ListNode * const check_point = check_point_allocated_blocks();
    const char** failed_names = (const char**)malloc(number_of_tests *
        sizeof(*failed_names));

```

```
void **current_state = NULL;
TestState group_state;

/* Find setup and teardown function */
for (i = 0; i < number_of_tests; i++) {
    const UnitTest * const test = &tests[i];

    if (test->function_type == UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP) {
        if (setup == NULL) {
            setup = test->function;
            setup_name = test->name;
            num_setups = 1;
        } else {
            print_error("[ ERROR ] More than one group setup function detected\n");
            exit_test(1);
        }
    }

    if (test->function_type == UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN) {
        if (teardown == NULL) {
            teardown = test->function;
            teardown_name = test->name;
            num_teardowns = 1;
        } else {
            print_error("[ ERROR ] More than one group teardown function detected\n");
            exit_test(1);
        }
    }
}
```

```
print_message("[=====] Running %\"PRIdS \" test(s).\n",
    number_of_tests - num_setups - num_teardowns);

if (setup != NULL) {
    int failed;

    group_state.check_point = check_point_allocated_blocks();
    current_state = &group_state.state;
    *current_state = NULL;
    failed = _run_test(setup_name,
        setup,
        current_state,
        UNIT_TEST_FUNCTION_TYPE_SETUP,
        group_state.check_point);

    if (failed) {
        failed_names[total_failed] = setup_name;
    }

    total_failed += failed;
    tests_executed++;
}

while (current_test < number_of_tests) {
    int run_test = 0;
    const UnitTest * const test = &tests[current_test++];
    if (test->function == NULL) {
        continue;
    }
}
```

```
switch (test->function_type) {  
    case UNIT_TEST_FUNCTION_TYPE_TEST:  
        run_test = 1;  
        break;  
    case UNIT_TEST_FUNCTION_TYPE_SETUP:  
    case UNIT_TEST_FUNCTION_TYPE_TEARDOWN:  
    case UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP:  
    case UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN:  
        break;  
    default:  
        print_error("Invalid unit test function type %d\n",  
                   test->function_type);  
        break;  
    }  
  
if (run_test) {  
    int failed;  
  
    failed = _run_test(test->name,  
                      test->function,  
                      current_state,  
                      test->function_type,  
                      NULL);  
    if (failed) {  
        failed_names[total_failed] = test->name;  
    }  
  
    total_failed += failed;  
}
```

```

    tests_executed++;

}

}

if (teardown != NULL) {
    int failed;

    failed = _run_test(teardown_name,
                       teardown,
                       current_state,
                       UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN,
                       group_state.check_point);

    if (failed) {
        failed_names[total_failed] = teardown_name;
    }

    total_failed += failed;
    tests_executed++;
}

print_message("[=====] %"PRIIdS " test(s) run.\n", tests_executed);
print_error("[ PASSED ] %"PRIIdS " test(s).\n", tests_executed - total_failed);

if (total_failed) {
    print_error("[ FAILED ] %"PRIIdS " test(s), listed below:\n", total_failed);
    for (i = 0; i < total_failed; i++) {
        print_error("[ FAILED ] %s\n", failed_names[i]);
    }
} else {

```

```
    print_error("\n %"PRIIdS " FAILED TEST(S)\n", total_failed);

}

free((void*)failed_names);

fail_if_blocks_allocated(check_point, "run_group_tests");

return (int)total_failed;
```

```
}
```

```
#include "config.h"
```

```
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <cmocka_private.h>

static void mock_test_a_called(void)
```

```
{
```

```
    function_called();
```

```
}
```

```
static void mock_test_b_called(void)
```

```
{
```

```
    function_called();
```

```
}
```

```
static void mock_test_c_called(void)
```

```
{
```

```
    function_called();  
}  
  
}
```

```
static void test_does_fail_for_unexpected_call(void **state)  
{  
    (void)state;  
    expect_function_call(mock_test_a_called);  
    expect_function_call(mock_test_a_called);  
  
    mock_test_a_called();  
    mock_test_a_called();  
    mock_test_a_called();  
}
```

```
static void test_does_fail_for_unmade_expected_call(void **state)  
{  
    (void)state;  
    expect_function_call(mock_test_a_called);  
    expect_function_call(mock_test_a_called);  
  
    mock_test_a_called();  
}
```

```
static void test_ordering_fails_out_of_order(void **state)  
{  
    (void)state;  
    expect_function_call(mock_test_a_called);  
    expect_function_call(mock_test_b_called);  
    expect_function_call(mock_test_a_called);
```

```
    mock_test_b_called();

}

static void test_ordering_fails_out_of_order_for_at_least_once_calls(void **state)
{
    (void)state;
    expect_function_call_any(mock_test_a_called);
    ignore_function_calls(mock_test_b_called);

    mock_test_b_called();
    mock_test_c_called();
}

/* Primarily used to test error message */

static void test_fails_out_of_order_if_no_calls_found_on_any(void **state)
{
    (void)state;
    expect_function_call_any(mock_test_a_called);
    ignore_function_calls(mock_test_b_called);

    mock_test_a_called();
    mock_test_c_called();
}

static void test_fails_if_zero_count_used(void **state)
{
    (void)state;
    expect_function_calls(mock_test_a_called, 0);
```

```
    mock_test_a_called();

}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_does_fail_for_unexpected_call)
        ,cmocka_unit_test(test_does_fail_for_unmade_expected_call)
        ,cmocka_unit_test(test_does_fail_for_unmade_expected_call)
        ,cmocka_unit_test(test_ordering_fails_out_of_order)
        ,cmocka_unit_test(test_ordering_fails_out_of_order_for_at_least_once_calls)
        ,cmocka_unit_test(test_fails_out_of_order_if_no_calls_found_on_any)
        ,cmocka_unit_test(test_fails_if_zero_count_used)
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
* See the License for the specific language governing permissions and  
* limitations under the License.
```

```
*/
```

```
#include <stdarg.h>
```

```
#include <stddef.h>
```

```
#include <setjmp.h>
```

```
#include <string.h>
```

```
#include <cmocka.h>
```

```
#include "key_value.h"
```

```
static KeyValue key_values[] = {  
    { 10, "this" },  
    { 52, "test" },  
    { 20, "a" },  
    { 13, "is" },  
};
```

```
static int create_key_values(void **state) {  
    KeyValue * const items = (KeyValue*)test_malloc(sizeof(key_values));  
    memcpy(items, key_values, sizeof(key_values));  
    *state = (void*)items;  
    set_key_values(items, sizeof(key_values) / sizeof(key_values[0]));  
  
    return 0;  
}
```

```
static int destroy_key_values(void **state) {  
    test_free(*state);
```

```

    set_key_values(NULL, 0);

    return 0;
}

static void test_find_item_by_value(void **state) {
    unsigned int i;

    (void) state; /* unused */

    for (i = 0; i < sizeof(key_values) / sizeof(key_values[0]); i++) {
        KeyValue * const found = find_item_by_value(key_values[i].value);
        assert_true(found != NULL);
        assert_int_equal(found->key, key_values[i].key);
        assert_string_equal(found->value, key_values[i].value);
    }
}

static void test_sort_items_by_key(void **state) {
    unsigned int i;
    KeyValue * const kv = *state;
    sort_items_by_key();
    for (i = 1; i < sizeof(key_values) / sizeof(key_values[0]); i++) {
        assert_true(kv[i - 1].key < kv[i].key);
    }
}

int main(void) {
    const struct CMUnitTests tests[] = {

```

```
    cmocka_unit_test_setup_teardown(test_find_item_by_value,
                                    create_key_values, destroy_key_values),
    cmocka_unit_test_setup_teardown(test_sort_items_by_key,
                                    create_key_values, destroy_key_values),
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <database.h>

extern DatabaseConnection* connect_to_product_database(void);
```

```
/* Mock connect to database function.

* NOTE: This mock function is very general could be shared between tests
* that use the imaginary database.h module. */

DatabaseConnection* connect_to_database(const char * const url,
                                         const unsigned int port) {

    check_expected_ptr(url);
    check_expected(port);
    return (DatabaseConnection*)((size_t)mock());
}

static void test_connect_to_product_database(void **state) {
    (void) state; /* unused */

    expect_string(connect_to_database, url, "products.abcd.org");
    expect_value(connect_to_database, port, 322);
    will_return(connect_to_database, 0xDA7ABA53);
    assert_int_equal((size_t)connect_to_product_database(), 0xDA7ABA53);
}

/* This test will fail since the expected URL is different to the URL that is
 * passed to connect_to_database() by connect_to_product_database(). */

static void test_connect_to_product_database_bad_url(void **state) {
    (void) state; /* unused */

    expect_string(connect_to_database, url, "products.abcd.com");
    expect_value(connect_to_database, port, 322);
    will_return(connect_to_database, 0xDA7ABA53);
    assert_int_equal((size_t)connect_to_product_database(), 0xDA7ABA53);
```

```
}
```

```
/* This test will fail since the mock connect_to_database() will attempt to
 * retrieve a value for the parameter port which isn't specified by this
 * test function. */

static void test_connect_to_product_database_missing_parameter(void **state) {
    (void) state; /* unused */

    expect_string(connect_to_database, url, "products.abcd.org");
    will_return(connect_to_database, 0xDA7ABA53);
    assert_int_equal((size_t)connect_to_product_database(), 0xDA7ABA53);
}
```

```
int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_connect_to_product_database),
        cmocka_unit_test(test_connect_to_product_database_bad_url),
        cmocka_unit_test(test_connect_to_product_database_missing_parameter),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0

```

```
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/
```

```
typedef struct KeyValue {  
    unsigned int key;  
    const char* value;  
} KeyValue;  
  
void set_key_values(KeyValue * const new_key_values,  
                    const unsigned int new_number_of_key_values);  
  
KeyValue* find_item_by_value(const char * const value);  
  
void sort_items_by_key(void);  
/*  
 * Copyright 2008 Google Inc.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 * http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software
```

```
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/
```

```
#include <assert.h>
```

```
#include "assert_module.h"
```

```
/* If unit testing is enabled override assert with mock_assert(). */  
#ifdef UNIT_TESTING  
extern void mock_assert(const int result, const char* const expression,  
                      const char * const file, const int line);  
  
#undef assert  
  
#define assert(expression) \  
    mock_assert(((expression) ? 1 : 0), #expression, __FILE__, __LINE__);  
#endif /* UNIT_TESTING */
```

```
void increment_value(int * const value) {  
    assert(value);  
    (*value)++;  
}
```

```
void decrement_value(int * const value) {  
    if (value) {  
        (*value) --;  
    }  
}
```

```
/*
```

```
* Copyright 2008 Google Inc.  
*  
* Licensed under the Apache License, Version 2.0 (the "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/
```

```
/* A calculator example used to demonstrate the cmocka testing library. */
```

```
#ifdef HAVE_CONFIG_H  
#include "config.h"  
#endif  
  
#include <assert.h>  
  
#ifdef HAVE_MALLOC_H  
#include <malloc.h>  
#endif  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>
```

```
/* If this is being built for a unit test. */
```

```
#ifdef UNIT_TESTING

/* Redirect printf to a function in the test application so it's possible to
 * test the standard output. */

#ifndef printf
#define printf example_test_printf

#ifndef print_message
#define print_message example_test_message
#endif

/* Redirect fprintf to a function in the test application so it's possible to
 * test error messages. */

#ifndef fprintf
#define fprintf example_test_fprintf

#ifndef example_test_fprintf
#define example_test_fprintf extern int example_test_fprintf(FILE * const file, const char *format, ...);
#endif
#endif

/* Redirect assert to mock_assert() so assertions can be caught by cmocka. */

#ifndef assert
#define assert(expression) \
    mock_assert((int)(expression), #expression, __FILE__, __LINE__)
#endif

void mock_assert(const int result, const char* expression, const char *file,
                const int line);
```

```

/* Redirect calloc and free to test_malloc() and test_free() so cmocka can
 * check for memory leaks. */

#ifndef malloc
#define malloc _test_malloc
#endif /* malloc */

#define free(_ptr) _test_free(_ptr)

void* _test_malloc(const size_t number_of_elements, const size_t size,
                  const char* file, const int line);
void _test_free(void* const ptr, const char* file, const int line);

int example_main(int argc, char *argv[]);

/* main is defined in the unit test so redefine name of the main function
 * here. */

#define main example_main

/* All functions in this object need to be exposed to the test application,
 * so redefine static to nothing. */

#define static

#endif /* UNIT_TESTING */

/* A binary arithmetic integer operation (add, subtract etc.) */

typedef int (*BinaryOperator)(int a, int b);

```

```

/* Structure which maps operator strings to functions. */
typedef struct OperatorFunction {
    const char* operator;
    BinaryOperator function;
} OperatorFunction;

BinaryOperator find_operator_function_by_string(
    const size_t number_of_operator_functions,
    const OperatorFunction * const operator_functions,
    const char* const operator_string);

int perform_operation(
    int number_of_arguments, char *arguments[],
    const size_t number_of_operator_functions,
    const OperatorFunction * const operator_functions,
    int * const number_of_intermediate_values,
    int ** const intermediate_values, int * const error_occurred);

static int add(int a, int b);
static int subtract(int a, int b);
static int multiply(int a, int b);
static int divide(int a, int b);

/* Associate operator strings to functions. */
static OperatorFunction operator_function_map[] = {
    {"+", add},
    {"-", subtract},

```

```

        {"*", multiply},
        {"/", divide},
    };

static int add(int a, int b) {
    return a + b;
}

static int subtract(int a, int b) {
    return a - b;
}

static int multiply(int a, int b) {
    return a * b;
}

static int divide(int a, int b) {
    assert(b); /* Check for divide by zero. */
    return a / b;
}

/* Searches the specified array of operator_functions for the function
 * associated with the specified operator_string. This function returns the
 * function associated with operator_string if successful, NULL otherwise.
 */
BinaryOperator find_operator_function_by_string(
    const size_t number_of_operator_functions,
    const OperatorFunction * const operator_functions,
    const char* const operator_string) {

```

```

size_t i;

assert(!number_of_operator_functions || operator_functions);
assert(operator_string != NULL);

for (i = 0; i < number_of_operator_functions; i++) {
    const OperatorFunction *const operator_function =
        &operator_functions[i];
    if (strcmp(operator_function->operator, operator_string) == 0) {
        return operator_function->function;
    }
}
return NULL;
}

/* Perform a series of binary arithmetic integer operations with no operator
 * precedence.
 *
 * The input expression is specified by arguments which is an array of
 * containing number_of_arguments strings. Operators invoked by the expression
 * are specified by the array operator_functions containing
 * number_of_operator_functions, OperatorFunction structures. The value of
 * each binary operation is stored in a pointer returned to intermediate_values
 * which is allocated by malloc().
 *
 * If successful, this function returns the integer result of the operations.
 * If an error occurs while performing the operation error_occurred is set to
 * 1, the operation is aborted and 0 is returned.
 */
int perform_operation(

```

```
int number_of_arguments, char *arguments[],  
const size_t number_of_operator_functions,  
const OperatorFunction * const operator_functions,  
int * const number_of_intermediate_values,  
int ** const intermediate_values, int * const error_occurred) {  
  
char *end_of_integer;  
  
int value;  
  
int i;  
  
assert(!number_of_arguments || arguments);  
assert(!number_of_operator_functions || operator_functions);  
assert(error_occurred != NULL);  
assert(number_of_intermediate_values != NULL);  
assert(intermediate_values != NULL);  
  
*error_occurred = 0;  
*number_of_intermediate_values = 0;  
*intermediate_values = NULL;  
  
if (!number_of_arguments)  
    return 0;  
  
/* Parse the first value. */  
  
value = (int)strtol(arguments[0], &end_of_integer, 10);  
if (end_of_integer == arguments[0]) {  
    /* If an error occurred while parsing the integer. */  
    fprintf(stderr, "Unable to parse integer from argument %s\n",  
        arguments[0]);  
    *error_occurred = 1;  
    return 0;  
}
```

```
/* Allocate an array for the output values. */
*intermediate_values = calloc(((number_of_arguments - 1) / 2),
                           sizeof(**intermediate_values));

i = 1;

while (i < number_of_arguments) {

    int other_value;

    const char* const operator_string = arguments[i];

    const BinaryOperator function = find_operator_function_by_string(
        number_of_operator_functions, operator_functions, operator_string);

    int * const intermediate_value =
        &(*intermediate_values)[*number_of_intermediate_values];

    (*number_of_intermediate_values)++;



    if (!function) {
        fprintf(stderr, "Unknown operator %s, argument %d\n",
                operator_string, i);
        *error_occurred = 1;
        break;
    }

    i++;



    if (i == number_of_arguments) {
        fprintf(stderr, "Binary operator %s missing argument\n",
                operator_string);
        *error_occurred = 1;
        break;
    }
}
```

```
other_value = (int)strtol(arguments[i], &end_of_integer, 10);

if (end_of_integer == arguments[i]) {
    /* If an error occurred while parsing the integer. */
    fprintf(stderr, "Unable to parse integer %s of argument %d\n",
            arguments[i], i);

    *error_occurred = 1;
    break;
}

i++;

/* Perform the operation and store the intermediate value. */
*intermediate_value = function(value, other_value);
value = *intermediate_value;

}

if (*error_occurred) {
    free(*intermediate_values);
    *intermediate_values = NULL;
    *number_of_intermediate_values = 0;
    return 0;
}

return value;
}

int main(int argc, char *argv[]) {
    int return_value;
    int number_of_intermediate_values;
    int *intermediate_values;
    /* Peform the operation. */
}
```

```
const int result = perform_operation(
    argc - 1, &argv[1],
    sizeof(operator_function_map) / sizeof(operator_function_map[0]),
    operator_function_map, &number_of_intermediate_values,
    &intermediate_values, &return_value);

/* If no errors occurred display the result.*/
if (!return_value && argc > 1) {
    int i;
    int intermediate_value_index = 0;
    printf("%s\n", argv[1]);
    for (i = 2; i < argc; i += 2) {
        assert(intermediate_value_index < number_of_intermediate_values);
        printf(" %s %s = %d\n", argv[i], argv[i + 1],
            intermediate_values[intermediate_value_index++]);
    }
    printf("= %d\n", result);
}

if (intermediate_values) {
    free(intermediate_values);
}

return return_value;
}

const char* get_status_code_string(const unsigned int status_code);
unsigned int string_to_status_code(const char* const status_code_string);
/*
 * Copyright 2008 Google Inc.
 *
```

```
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
```

```
void increment_value(int * const value);
void decrement_value(int * const value);
/*
* Copyright 2008 Google Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
```

```
*/  
typedef struct DatabaseConnection DatabaseConnection;  
  
/* Function that takes an SQL query string and sets results to an array of  
* pointers with the result of the query. The value returned specifies the  
* number of items in the returned array of results. The returned array of  
* results are statically allocated and should not be deallocated using free()  
*/  
typedef unsigned int (*QueryDatabase)(  
    DatabaseConnection* const connection, const char * const query_string,  
    void *** const results);  
  
/* Connection to a database. */  
struct DatabaseConnection {  
    const char *url;  
    unsigned int port;  
    QueryDatabase query_database;  
};  
  
/* Connect to a database. */  
DatabaseConnection* connect_to_database(const char * const url,  
                                      const unsigned int port);  
  
/*  
 * Copyright 2013 (c) Andreas Schneider <asn@cynapses.org>  
 *          Jakub Hrozek <jakub.hrozek@gmail.com>  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 */
```

```
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.

*/
```

```
int chef_cook(const char *order, char **dish_out);

const char *chef_strerror(int error);

/*
* Copyright 2013 (c) Andreas Schneider <asn@cynapses.org>
* Jakub Hrozek <jakub.hrozek@gmail.com>
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.

*/
```

```
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <stdio.h>
#include <errno.h>
#include <stdbool.h>
#include <string.h>

#include "chef.h"

/* This is the real chef, just not implemented yet, currently it always
 * returns ENOSYS
 */
int chef_cook(const char *order, char **dish_out)
{
    if (order == NULL || dish_out == NULL) return EINVAL;

    return -ENOSYS;
}

/* Print chef return codes as string */
const char *chef_strerror(int error)
{
    switch (error) {
        case 0:
            return "Success";
    }
}
```

```
case -1:  
    return "Unknown dish";  
  
case -2:  
    return "Not enough ingredients for the dish";  
  
}  
  
return "Unknown error!";  
}
```

```
int __wrap_chef_cook(const char *order, char **dish_out);  
/*  
 * Copyright 2013 (c) Andreas Schneider <asn@cynapses.org>  
 * Jakub Hrozek <jakub.hrozek@gmail.com>  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * you may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 * http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */
```

```
#include <errno.h>
```

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

#include "waiter_test_wrap.h"
#include "chef.h"

/*
 * This is a mocked Chef object. A real Chef would look if he knows
 * the dish in some kind of internal database and check his storage for
 * ingredients. This chef simply retrieves this information from the test
 * that is calling him.
 *
 * This object is also wrapped - if any code links with this file and is
 * compiled with linker option --wrap chef_cook, any calls of that code to
 * chef_cook will end up calling __wrap_chef_cook.
 *
 * If for any reason the wrapped function wanted to call the real chef_cook()
 * function, it could do so by calling the special symbol __real_chef_cook().
 *
 * Please note that when setting return codes for the chef_cook function, we
 * use this wrapper as a parameter for the will_return() macro, not the
 * real function.
```

```
*  
* A chef object would return:  
* 0 - cooking dish went fine  
* -1 - unknown dish  
* -2 - ran out of ingredients for the dish  
* any other error code -- unexpected error while cooking  
*  
* The return codes should be consistent between the real and mocked objects.  
*/  
  
int __wrap_chef_cook(const char *order, char **dish_out)  
{  
    bool has_ingredients;  
    bool knows_dish;  
    char *dish;  
  
    check_expected_ptr(order);  
  
    knows_dish = mock_type(bool);  
    if (knows_dish == false) {  
        return -1;  
    }  
  
    has_ingredients = mock_type(bool);  
    if (has_ingredients == false) {  
        return -2;  
    }  
  
    dish = mock_ptr_type(char *);  
    *dish_out = strdup(dish);
```

```
if (*dish_out == NULL) return ENOMEM;

return mock_type(int);

}

/* Waiter return codes:
 * 0 - success
 * -1 - kitchen failed
 * -2 - kitchen succeeded, but cooked a different food
 */

static int waiter_process(const char *order, char **dish)
{
    int rv;

    rv = chef_cook(order, dish);

    if (rv != 0) {
        fprintf(stderr, "Chef couldn't cook %s: %s\n",
                order, chef_strerror(rv));
        return -1;
    }

    /* Check if we received the dish we wanted from the kitchen */
    if (strcmp(order, *dish) != 0) {
        free(*dish);
        *dish = NULL;
        return -2;
    }

    return 0;
}
```

```
}
```

```
static void test_order_hotdog(void **state)
{
    int rv;
    char *dish;

    (void) state; /* unused */

    /* We expect the chef to receive an order for a hotdog */
    expect_string(__wrap_chef_cook, order, "hotdog");
    /* And we tell the test chef that ke knows how to cook a hotdog
     * and has the ingredients
     */
    will_return(__wrap_chef_cook, true);
    will_return(__wrap_chef_cook, true);
    /* The result will be a hotdog and the cooking process will succeed */
    will_return(__wrap_chef_cook, cast_ptr_to_largest_integral_type("hotdog"));
    will_return(__wrap_chef_cook, 0);

    /* Test the waiter */
    rv = waiter_process("hotdog", &dish);

    /* We expect the cook to succeed cooking the hotdog */
    assert_int_equal(rv, 0);
    /* And actually receive one */
    assert_string_equal(dish, "hotdog");
    if (dish != NULL) {
        free(dish);
    }
}
```

```
    }

}

static void test_bad_dish(void **state)
{
    int rv;
    char *dish;

    (void) state; /* unused */

    /* We expect the chef to receive an order for a hotdog */
    expect_string(__wrap_chef_cook, order, "hotdog");

    /* And we tell the test chef that ke knows how to cook a hotdog
     * and has the ingredients
     */
    will_return(__wrap_chef_cook, true);

    will_return(__wrap_chef_cook, true);

    /* The result will be a burger and the cooking process will succeed.
     * We expect the waiter to handle the bad dish and return an error
     * code
     */
    will_return(__wrap_chef_cook, cast_ptr_to_largest_integral_type("burger"));

    will_return(__wrap_chef_cook, 0);

    /* Test the waiter */
    rv = waiter_process("hotdog", &dish);

    /* According to the documentation the waiter should return -2 now */
    assert_int_equal(rv, -2);
```

```
/* And do not give the bad dish to the customer */
assert_null(dish);

}

int main(void)
{
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_order_hotdog),
        cmocka_unit_test(test_bad_dish),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/

```

```
#include <string.h>
#include "assert_macro.h"

static const char* status_code_strings[] = {
    "Address not found",
    "Connection dropped",
    "Connection timed out",
};

const char* get_status_code_string(const unsigned int status_code) {
    return status_code_strings[status_code];
}

unsigned int string_to_status_code(const char* const status_code_string) {
    unsigned int i;
    for (i = 0; i < sizeof(status_code_strings) /
        sizeof(status_code_strings[0]); i++) {
        if (strcmp(status_code_strings[i], status_code_string) == 0) {
            return i;
        }
    }
    return ~0U;
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at

```

```
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/  
  
#include <stdarg.h>  
#include <stddef.h>  
#include <setjmp.h>  
#include <cmocka.h>  
#include <database.h>  
  
  
extern DatabaseConnection* connect_to_customer_database();  
extern unsigned int get_customer_id_by_name(  
    DatabaseConnection * const connection, const char * const customer_name);  
  
  
/* Mock query database function. */  
static unsigned int mock_query_database(DatabaseConnection* const connection,  
                                       const char * const query_string,  
                                       void *** const results) {  
    (void) connection; /* unused */  
    (void) query_string; /* unused */  
  
    *results = (void **)mock_ptr_type(int *);  
    return mock_ptr_type(int);  
}
```

```

/* Mock of the connect to database function. */

DatabaseConnection* connect_to_database(const char * const database_url,
                                       const unsigned int port) {
    (void) database_url; /* unused */
    (void) port; /* unused */

    return (DatabaseConnection*)((size_t)mock());
}

static void test_connect_to_customer_database(void **state) {
    (void) state; /* unused */

    will_return(connect_to_database, 0x0DA7ABA53);

    assert_int_equal((size_t)connect_to_customer_database(), 0x0DA7ABA53);
}

/* This test fails as the mock function connect_to_database() will have no
 * value to return. */

#ifndef O
static void fail_connect_to_customer_database(void **state) {
    (void) state; /* unused */

    assert_true(connect_to_customer_database() ==
               (DatabaseConnection*)0x0DA7ABA53);
}
#endif

```

```
static void test_get_customer_id_by_name(void **state) {
    DatabaseConnection connection = {
        "somedatabase.somewhere.com", 12345678, mock_query_database
    };
    /* Return a single customer ID when mock_query_database() is called. */
    int customer_ids = 543;
    int rc;

    (void) state; /* unused */

    will_return(mock_query_database,
               cast_ptr_to_largest_integral_type(&customer_ids));
    will_return(mock_query_database, 1);

    rc = get_customer_id_by_name(&connection, "john doe");
    assert_int_equal(rc, 543);
}

int main(void) {
    const struct CMUnitTests tests[] = {
        cmocka_unit_test(test_connect_to_customer_database),
        cmocka_unit_test(test_get_customer_id_by_name),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");

```

```
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.
```

```
*/
```

```
#include <stdarg.h>
```

```
#include <stddef.h>
```

```
#include <setjmp.h>
```

```
#include <cmocka.h>
```

```
extern void leak_memory();
```

```
extern void buffer_overflow();
```

```
extern void buffer_underflow();
```

```
/* Test case that fails as leak_memory() leaks a dynamically allocated block. */
```

```
static void leak_memory_test(void **state) {
```

```
    (void) state; /* unused */
```

```
    leak_memory();
```

```
}
```

```
/* Test case that fails as buffer_overflow() corrupts an allocated block. */
```

```
static void buffer_overflow_test(void **state) {
```

```
(void) state; /* unused */

buffer_overflow();

}

/* Test case that fails as buffer_underflow() corrupts an allocated block. */

static void buffer_underflow_test(void **state) {
    (void) state; /* unused */

    buffer_underflow();

}

int main(void) {
    const struct CMUnitTests tests[] = {
        cmocka_unit_test(leak_memory_test),
        cmocka_unit_test(buffer_overflow_test),
        cmocka_unit_test(buffer_underflow_test),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 *
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 */
```

* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

* See the License for the specific language governing permissions and
* limitations under the License.

*/

```
#ifdef HAVE_CONFIG_H
```

```
#include "config.h"
```

```
#endif
```

```
#ifdef HAVE_MALLOC_H
```

```
#include <malloc.h>
```

```
#endif
```

```
#include <sys/types.h>
```

```
#include <stdlib.h>
```

```
#ifdef UNIT_TESTING
```

```
extern void* _test_malloc(const size_t size, const char* file, const int line);
```

```
extern void* _test_calloc(const size_t number_of_elements, const size_t size,  
const char* file, const int line);
```

```
extern void _test_free(void* const ptr, const char* file, const int line);
```

```
#define malloc(size) _test_malloc(size, __FILE__, __LINE__)
```

```
#define calloc(num, size) _test_calloc(num, size, __FILE__, __LINE__)
```

```
#define free(ptr) _test_free(ptr, __FILE__, __LINE__)
```

```
#endif // UNIT_TESTING
```

```
void leak_memory(void);
```

```
void buffer_overflow(void);
```

```
void buffer_underflow(void);
```

```
void leak_memory(void) {
    int * const temporary = (int*)malloc(sizeof(int));
    *temporary = 0;
}

void buffer_overflow(void) {
    char * const memory = (char*)malloc(sizeof(int));
    memory[sizeof(int)] = '!';
    free(memory);
}

void buffer_underflow(void) {
    char * const memory = (char*)malloc(sizeof(int));
    memory[-1] = '!';
    free(memory);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```

* See the License for the specific language governing permissions and
* limitations under the License.

*/
#include <stddef.h>
#include <stdlib.h>
#include <string.h>

#include "key_value.h"

static KeyValue *key_values = NULL;
static unsigned int number_of_key_values = 0;

void set_key_values(KeyValue * const new_key_values,
                    const unsigned int new_number_of_key_values) {
    key_values = new_key_values;
    number_of_key_values = new_number_of_key_values;
}

/* Compare two key members of KeyValue structures. */
static int key_value_compare_keys(const void *a, const void *b) {
    return (int)((KeyValue*)a)->key - (int)((KeyValue*)b)->key;
}

/* Search an array of key value pairs for the item with the specified value. */
KeyValue* find_item_by_value(const char * const value) {
    unsigned int i;
    for (i = 0; i < number_of_key_values; i++) {
        if (strcmp(key_values[i].value, value) == 0) {
            return &key_values[i];
        }
    }
}

```

```
    }

}

return NULL;

}

/* Sort an array of key value pairs by key. */

void sort_items_by_key(void) {

    qsort(key_values, number_of_key_values, sizeof(*key_values),
          key_value_compare_keys);

}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <database.h>

DatabaseConnection* connect_to_product_database(void);
```

```
/* Connect to the database containing customer information. */

DatabaseConnection* connect_to_product_database(void) {
    return connect_to_database("products.abcd.org", 322);
}

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

/* A test case that does nothing and succeeds. */
static void null_test_success(void **state) {
    (void) state; /* unused */
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(null_test_success),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 *
 * You may obtain a copy of the License at
 *
 *
```

```
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

#include "assert_module.h"

extern void increment_value(int * const value);

/* This test case will fail but the assert is caught by run_tests() and the
 * next test is executed. */
static void increment_value_fail(void **state) {
    (void) state;

    increment_value(NULL);
}

/* This test case succeeds since increment_value() asserts on the NULL
 * pointer. */
static void increment_value_assert(void **state) {
    (void) state;
```

```
    expect_assert_failure(increment_value(NULL));

}

/* This test case fails since decrement_value() doesn't assert on a NULL
 * pointer. */

static void decrement_value_fail(void **state) {
    (void) state;

    expect_assert_failure(decrement_value(NULL));

}

int main(void) {
    const struct CMUnitTests tests[] = {
        cmocka_unit_test(increment_value_fail),
        cmocka_unit_test(increment_value_assert),
        cmocka_unit_test(decrement_value_fail),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 *
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 */
```

```
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.

*/
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

#include "assert_macro.h"

/* This test will fail since the string returned by get_status_code_string(0)
 * doesn't match "Connection timed out". */
static void get_status_code_string_test(void **state) {
    (void) state; /* unused */

    assert_string_equal(get_status_code_string(0), "Address not found");
    assert_string_equal(get_status_code_string(1), "Connection timed out");
}

/* This test will fail since the status code of "Connection timed out" isn't 1 */
static void string_to_status_code_test(void **state) {
    (void) state; /* unused */

    assert_int_equal(string_to_status_code("Address not found"), 0);
    assert_int_equal(string_to_status_code("Connection timed out"), 1);
}
```

```
int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(get_status_code_string_test),
        cmocka_unit_test(string_to_status_code_test),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <stddef.h>
#include <stdio.h>
#include <database.h>
#ifndef _WIN32
#define sprintf _snprintf
#endif /* _WIN32 */
```

```

DatabaseConnection* connect_to_customer_database(void);

unsigned int get_customer_id_by_name(
    DatabaseConnection * const connection,
    const char * const customer_name);

/* Connect to the database containing customer information. */

DatabaseConnection* connect_to_customer_database(void) {
    return connect_to_database("customers.abcd.org", 321);
}

/* Find the ID of a customer by his/her name returning a value > 0 if
 * successful, 0 otherwise. */

unsigned int get_customer_id_by_name(
    DatabaseConnection * const connection,
    const char * const customer_name) {
    char query_string[256];
    int number_of_results;
    void **results;
    snprintf(query_string, sizeof(query_string),
        "SELECT ID FROM CUSTOMERS WHERE NAME = %s", customer_name);
    number_of_results = connection->query_database(connection, query_string,
        &results);

    if (number_of_results != 1) {
        return -1;
    }

    return (unsigned int)*((int *)results);
}

```

```
}

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include "cmocka.h"
#include <stdio.h>

#ifndef _WIN32
/* Compatibility with the Windows standard C library. */
#define vsnprintf _vsnprintf
#endif /* _WIN32 */

#define array_length(x) (sizeof(x) / sizeof((x)[0]))
```

```
/* To simplify this code, these functions and data structures could have been
 * separated out from the application example.c into a header shared with
 * test application. However, this example illustrates how it's possible to
 * test existing code with little modification. */
```

```
typedef int (*BinaryOperator)(int a, int b);
```

```
typedef struct OperatorFunction {
    const char* operator;
    BinaryOperator function;
} OperatorFunction;
```

```
extern int add(int a, int b);
extern int subtract(int a, int b);
extern int multiply(int a, int b);
extern int divide(int a, int b);
extern BinaryOperator find_operator_function_by_string(
    const size_t number_of_operator_functions,
    const OperatorFunction * const operator_functions,
    const char* const operator_string);
extern int perform_operation(
    int number_of_arguments, char *arguments[],
    const size_t number_of_operator_functions,
    const OperatorFunction * const operator_functions,
    int * const number_of_intermediate_values,
    int ** const intermediate_values, int * const error_occurred);
extern int example_main(int argc, char *argv[]);
```

```
int example_test_fprintf(FILE* const file, const char *format, ...) CMOCKA_PRINTF_ATTRIBUTE(2, 3);
```

```
int example_test_printf(const char *format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);

static char temporary_buffer[256];

/* A mock fprintf function that checks the value of strings printed to the
 * standard error stream. */

int example_test_fprintf(FILE* const file, const char *format, ...) {
    int return_value;
    va_list args;
    assert_true(file == stderr);
    va_start(args, format);
    return_value = vsnprintf(temporary_buffer, sizeof(temporary_buffer),
                           format, args);
    check_expected_ptr(temporary_buffer);
    va_end(args);
    return return_value;
}

/* A mock printf function that checks the value of strings printed to the
 * standard output stream. */

int example_test_printf(const char *format, ...) {
    int return_value;
    va_list args;
    va_start(args, format);
    return_value = vsnprintf(temporary_buffer, sizeof(temporary_buffer),
                           format, args);
    check_expected_ptr(temporary_buffer);
    va_end(args);
    return return_value;
}
```

```
}
```

```
/* A mock binary operator function. */
```

```
static int binary_operator(int a, int b) {
```

```
    check_expected(a);
```

```
    check_expected(b);
```

```
    return (int)mock();
```

```
}
```

```
/* Ensure add() adds two integers correctly. */
```

```
static void test_add(void **state) {
```

```
    (void) state; /* unused */
```

```
    assert_int_equal(add(3, 3), 6);
```

```
    assert_int_equal(add(3, -3), 0);
```

```
}
```

```
/* Ensure subtract() subtracts two integers correctly. */
```

```
static void test_subtract(void **state) {
```

```
    (void) state; /* unused */
```

```
    assert_int_equal(subtract(3, 3), 0);
```

```
    assert_int_equal(subtract(3, -3), 6);
```

```
}
```

```
/* Ensure multiple() multiplies two integers correctly. */
```

```
static void test_multiply(void **state) {
```

```
    (void) state; /* unused */
```

```
    assert_int_equal(multiply(3, 3), 9);
    assert_int_equal(multiply(3, 0), 0);
}

/* Ensure divide() divides one integer by another correctly. */
static void test_divide(void **state) {
    (void) state; /* unused */

    assert_int_equal(divide(10, 2), 5);
    assert_int_equal(divide(2, 10), 0);
}

/* Ensure divide() asserts when trying to divide by zero. */
static void test_divide_by_zero(void **state) {
    (void) state; /* unused */

    expect_assert_failure(divide(100, 0));
}

/* Ensure find_operator_function_by_string() asserts when a NULL pointer is
 * specified as the table to search. */
static void test_find_operator_function_by_string_null_functions(void **state) {
    (void) state; /* unused */

    expect_assert_failure(find_operator_function_by_string(1, NULL, "test"));
}

/* Ensure find_operator_function_by_string() asserts when a NULL pointer is
```

```

/* specified as the string to search for. */

static void test_find_operator_function_by_string_null_string(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
    };

    (void) state; /* unused */

    expect_assert_failure(find_operator_function_by_string(
        array_length(operator_functions), operator_functions, NULL));
}

/* Ensure find_operator_function_by_string() returns NULL when a NULL pointer
 * is specified as the table to search when the table size is 0. */

static void test_find_operator_function_by_string_valid_null_functions(void **state) {
    (void) state; /* unused */

    assert_null(find_operator_function_by_string(0, NULL, "test"));
}

/* Ensure find_operator_function_by_string() returns NULL when searching for
 * an operator string that isn't in the specified table. */

static void test_find_operator_function_by_string_not_found(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
        {"-", binary_operator},
        {"/", binary_operator},
    };
}

```

```
(void) state; /* unused */

assert_null(find_operator_function_by_string(
    array_length(operator_functions), operator_functions, "test"));

}
```

```
/* Ensure find_operator_function_by_string() returns the correct function when
 * searching for an operator string that is in the specified table. */

static void test_find_operator_function_by_string_found(void **state) {

    const OperatorFunction operator_functions[] = {

        {"+", (BinaryOperator)0x12345678},
        {"-", (BinaryOperator)0xDEADBEEF},
        {"/", (BinaryOperator)0xABADCAFE},
    };

}
```

```
(void) state; /* unused */

assert_int_equal(
    cast_ptr_to_largest_integral_type(
        find_operator_function_by_string(array_length(operator_functions),
            operator_functions,
            "-")),
    0xDEADBEEF);

}
```

```
/* Ensure perform_operation() asserts when a NULL arguments array is specified. */

static void test_perform_operation_null_args(void **state) {

    const OperatorFunction operator_functions[] = {

        {"+", binary_operator},
    }
```

```

};

int number_of_intermediate_values;
int *intermediate_values;
int error_occurred;

(void) state; /* unused */

expect_assert_failure(perform_operation(
    1, NULL, array_length(operator_functions), operator_functions,
    &number_of_intermediate_values, &intermediate_values,
    &error_occurred));
}

/* Ensure perform_operation() asserts when a NULL operator_functions array is
 * specified. */
static void test_perform_operation_null_operator_functions(void **state) {
    const char *args[] = {
        "1", "+", "2", "*", "4"
    };
    int number_of_intermediate_values;
    int *intermediate_values;
    int error_occurred;

    (void) state; /* unused */

    expect_assert_failure(perform_operation(
        array_length(args), (char **) args, 1, NULL, &number_of_intermediate_values,
        &intermediate_values, &error_occurred));
}

```

```

/* Ensure perform_operation() asserts when a NULL pointer is specified for
 * number_of_intermediate_values. */

static void test_perform_operation_null_number_of_intermediate_values(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
    };

    const char *args[] = {
        "1", "+", "2", "*", "4"
    };

    int *intermediate_values;
    int error_occurred;

    (void) state; /* unused */

    expect_assert_failure(perform_operation(
        array_length(args), (char **) args, 1, operator_functions, NULL,
        &intermediate_values, &error_occurred));
}

/* Ensure perform_operation() asserts when a NULL pointer is specified for
 * intermediate_values. */

static void test_perform_operation_null_intermediate_values(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
    };

    const char *args[] = {
        "1", "+", "2", "*", "4"
    };
}

```

```
int number_of_intermediate_values;
int error_occurred;

(void) state; /* unused */

expect_assert_failure(perform_operation(
    array_length(args), (char **) args, array_length(operator_functions),
    operator_functions, &number_of_intermediate_values, NULL,
    &error_occurred));

}

/* Ensure perform_operation() returns 0 when no arguments are specified. */
static void test_perform_operation_no_arguments(void **state) {
    int number_of_intermediate_values;
    int *intermediate_values;
    int error_occurred;

    (void) state; /* unused */

    assert_int_equal(perform_operation(
        0, NULL, 0, NULL, &number_of_intermediate_values, &intermediate_values,
        &error_occurred), 0);
    assert_int_equal(error_occurred, 0);

}

/* Ensure perform_operation() returns an error if the first argument isn't
 * an integer string. */
static void test_perform_operation_first_arg_not_integer(void **state) {
    const OperatorFunction operator_functions[] = {
```

```
        {"+", binary_operator},  
    };  
  
    const char *args[] = {  
        "test", "+", "2", "*", "4"  
    };  
  
    int number_of_intermediate_values;  
    int *intermediate_values;  
    int error_occurred;  
  
    (void) state; /* unused */  
  
    expect_string(example_test_fprintf, temporary_buffer,  
        "Unable to parse integer from argument test\n");  
  
    assert_int_equal(perform_operation(  
        array_length(args), (char **) args, array_length(operator_functions),  
        operator_functions, &number_of_intermediate_values,  
        &intermediate_values, &error_occurred), 0);  
    assert_int_equal(error_occurred, 1);  
}  
  
/* Ensure perform_operation() returns an error when parsing an unknown  
 * operator. */  
  
static void test_perform_operation_unknown_operator(void **state) {  
    const OperatorFunction operator_functions[] = {  
        {"+", binary_operator},  
    };  
  
    const char *args[] = {  
        "1", "*", "2", "*", "4"  
    };
```

```
};

int number_of_intermediate_values;
int *intermediate_values;
int error_occurred;

(void) state; /* unused */

expect_string(example_test_fprintf, temporary_buffer,
    "Unknown operator *, argument 1\n");

assert_int_equal(perform_operation(
    array_length(args), (char **) args, array_length(operator_functions),
    operator_functions, &number_of_intermediate_values,
    &intermediate_values, &error_occurred), 0);
assert_int_equal(error_occurred, 1);

}

/* Ensure perform_operation() returns an error when nothing follows an
 * operator. */

static void test_perform_operation_missing_argument(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
    };
    const char *args[] = {
        "1", "+",
    };
    int number_of_intermediate_values;
    int *intermediate_values;
    int error_occurred;
```

```
(void) state; /* unused */

expect_string(example_test_fprintf, temporary_buffer,
    "Binary operator + missing argument\n");

assert_int_equal(perform_operation(
    array_length(args), (char **) args, array_length(operator_functions),
    operator_functions, &number_of_intermediate_values,
    &intermediate_values, &error_occurred), 0);

assert_int_equal(error_occurred, 1);

}

/* Ensure perform_operation() returns an error when an integer doesn't follow
 * an operator. */

static void test_perform_operation_no_integer_after_operator(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
    };

    const char *args[] = {
        "1", "+", "test",
    };

    int number_of_intermediate_values;
    int *intermediate_values;
    int error_occurred;

    (void) state; /* unused */

    expect_string(example_test_fprintf, temporary_buffer,
```

```

    "Unable to parse integer test of argument 2\n");

assert_int_equal(perform_operation(
    array_length(args), (char **) args, array_length(operator_functions),
    operator_functions, &number_of_intermediate_values,
    &intermediate_values, &error_occurred), 0);

assert_int_equal(error_occurred, 1);

}

/* Ensure perform_operation() succeeds given valid input parameters. */
static void test_perform_operation(void **state) {
    const OperatorFunction operator_functions[] = {
        {"+", binary_operator},
        {"*", binary_operator},
    };

    const char *args[] = {
        "1", "+", "3", "*", "10",
    };

    int number_of_intermediate_values;
    int *intermediate_values = NULL;
    int error_occurred;

    (void) state; /* unused */

    /* Setup return values of mock operator functions. */
    /* Addition. */
    expect_value(binary_operator, a, 1);
    expect_value(binary_operator, b, 3);
}

```

```
    will_return(binary_operator, 4);

    /* Multiplication. */
    expect_value(binary_operator, a, 4);
    expect_value(binary_operator, b, 10);
    will_return(binary_operator, 40);

    assert_int_equal(perform_operation(
        array_length(args), (char **) args, array_length(operator_functions),
        operator_functions, &number_of_intermediate_values,
        &intermediate_values, &error_occurred), 40);
    assert_int_equal(error_occurred, 0);

    assert_non_null(intermediate_values);
    assert_int_equal(intermediate_values[0], 4);
    assert_int_equal(intermediate_values[1], 40);
    test_free(intermediate_values);
}

/* Ensure main() in example.c succeeds given no arguments. */
static void test_example_main_no_args(void **state) {
    const char *args[] = {
        "example",
    };

    (void) state; /* unused */

    assert_int_equal(example_main(array_length(args), (char **) args), 0);
```

```
}
```

```
/* Ensure main() in example.c succeeds given valid input arguments. */
static void test_example_main(void **state) {
    const char *args[] = {
        "example", "1", "+", "3", "*", "10",
    };

    (void) state; /* unused */

    expect_string(example_test_printf, temporary_buffer, "1\n");
    expect_string(example_test_printf, temporary_buffer, " + 3 = 4\n");
    expect_string(example_test_printf, temporary_buffer, " * 10 = 40\n");
    expect_string(example_test_printf, temporary_buffer, "= 40\n");

    assert_int_equal(example_main(array_length(args), (char **) args), 0);
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_add),
        cmocka_unit_test(test_subtract),
        cmocka_unit_test(test_multiply),
        cmocka_unit_test(test_divide),
        cmocka_unit_test(test_divide_by_zero),
        cmocka_unit_test(test_find_operator_function_by_string_null_functions),
    };
}
```

```
    cmocka_unit_test(test_find_operator_function_by_string_null_string),
    cmocka_unit_test(test_find_operator_function_by_string_valid_null_functions),
    cmocka_unit_test(test_find_operator_function_by_string_not_found),
    cmocka_unit_test(test_find_operator_function_by_string_found),
    cmocka_unit_test(test_perform_operation_null_args),
    cmocka_unit_test(test_perform_operation_null_operator_functions),
    cmocka_unit_test(test_perform_operation_null_number_of_intermediate_values),
    cmocka_unit_test(test_perform_operation_null_intermediate_values),
    cmocka_unit_test(test_perform_operation_no_arguments),
    cmocka_unit_test(test_perform_operation_first_arg_not_integer),
    cmocka_unit_test(test_perform_operation_unknown_operator),
    cmocka_unit_test(test_perform_operation_missing_argument),
    cmocka_unit_test(test_perform_operation_no_integer_after_operator),
    cmocka_unit_test(test_perform_operation),
    cmocka_unit_test(test_example_main_no_args),
    cmocka_unit_test(test_example_main),
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

/*
 * Copyright 2014 Luis Pabon, Jr.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
```

```
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/
```

```
/*  
* Programming by Contract is a programming methodology  
* which binds the caller and the function called to a  
* contract. The contract is represented using Hoare Triple:  
* {P} C {Q}  
* where {P} is the precondition before executing command C,  
* and {Q} is the postcondition.
```

```
*
```

```
* See also:
```

```
* http://en.wikipedia.org/wiki/Design\_by\_contract  
* http://en.wikipedia.org/wiki/Hoare\_logic  
* http://dlang.org/dbc.html
```

```
*/
```

```
#ifndef CMOCKA_PBC_H_  
#define CMOCKA_PBC_H_
```

```
#if defined(UNIT_TESTING) || defined (DEBUG)
```

```
#include <assert.h>
```

```
/*  
* Checks caller responsibility against contract  
*/
```

```
#define REQUIRE(cond) assert(cond)

/*
 * Checks function responsibility against contract.
 */

#define ENSURE(cond) assert(cond)

/*
 * While REQUIRE and ENSURE apply to functions, INVARIANT
 * applies to classes/structs. It ensures that instances
 * of the class/struct are consistent. In other words,
 * that the instance has not been corrupted.
 */

#define INVARIANT(invariant_fnc) do{ (invariant_fnc) } while (0);

#else

#define REQUIRE(cond) do { } while (0);
#define ENSURE(cond) do { } while (0);
#define INVARIANT(invariant_fnc) do{ } while (0);

#endif /* defined(UNIT_TESTING) || defined (DEBUG) */

#endif /* CMOCKA_PBC_H_ */

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at

```

```
*  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing, software  
* distributed under the License is distributed on an "AS IS" BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
* See the License for the specific language governing permissions and  
* limitations under the License.  
*/  
#ifndef CMOCKA_H_  
#define CMOCKA_H_  
  
#ifdef _WIN32  
# ifdef _MSC_VER  
  
#define __func__ __FUNCTION__  
  
# ifndef inline  
# define inline __inline  
# endif /* inline */  
  
# if _MSC_VER < 1500  
# ifdef __cplusplus  
extern "C" {  
# endif /* __cplusplus */  
int __stdcall IsDebuggerPresent();  
# ifdef __cplusplus  
} /* extern "C" */  
# endif /* __cplusplus */
```

```
# endif /* _MSC_VER < 1500 */

#ifndef _MSC_VER
#define _WIN32

/** @defgroup cmocka The CMocka API

 * These headers or their equivalents should be included prior to including
 * this header file.

 * @code
 * #include <stdarg.h>
 * #include <stddef.h>
 * #include <setjmp.h>
 * @endcode
 *
 * This allows test applications to use custom definitions of C standard
 * library functions and types.

 *
 * @{
 */

/* If __WORDSIZE is not set, try to figure it out and default to 32 bit. */
#ifndef __WORDSIZE
#if defined(__x86_64__) && !defined(__ILP32__)
#define __WORDSIZE 64
#else
#define __WORDSIZE 32
#endif
#endif
```

```
#ifdef DOXYGEN

/** 
 * Largest integral type. This type should be large enough to hold any
 * pointer or integer supported by the compiler.
 */

typedef uintmax_t LargestIntegralType;

#else /* DOXGEN */

#ifndef LargestIntegralType
# if __WORDSIZE == 64
# define LargestIntegralType unsigned long int
# else
# define LargestIntegralType unsigned long long int
# endif
#endif /* LargestIntegralType */

#endif /* DOXYGEN */

/* Printf format used to display LargestIntegralType as a hexadecimal. */
#ifndef LargestIntegralTypePrintfFormat
# ifdef _WIN32
# define LargestIntegralTypePrintfFormat "0x%I64x"
# else
# if __WORDSIZE == 64
# define LargestIntegralTypePrintfFormat "%#lx"
# else
# define LargestIntegralTypePrintfFormat "%#llx"
# endif
# endif
#endif /* _WIN32 */
#endif /* LargestIntegralTypePrintfFormat */
```

```
/* Printf format used to display LargestIntegralType as a decimal. */

#ifndef LargestIntegralTypePrintfFormatDecimal
#define _WIN32
#define LargestIntegralTypePrintfFormatDecimal "%I64u"
#else
# if __WORDSIZE == 64
#define LargestIntegralTypePrintfFormatDecimal "%lu"
#else
#define LargestIntegralTypePrintfFormatDecimal "%llu"
#endif
#endif /* _WIN32 */
#endif /* LargestIntegralTypePrintf */
```

```
/* Perform an unsigned cast to LargestIntegralType. */

#define cast_to_largest_integral_type(value) \
((LargestIntegralType)(value))

/* Smallest integral type capable of holding a pointer. */

#if !defined(_UINTPTR_T) && !defined(_UINTPTR_T_DEFINED)
#if defined(_WIN32)
/* WIN32 is an ILP32 platform */
typedef unsigned int uintptr_t;
#elif defined(_WIN64)
typedef unsigned long int uintptr_t
#endif /* _WIN32 */
#endif /* _UINTPTR_T */
```

```
/* ILP32 and LP64 platforms */

#endif /* __WORDSIZE */
```

```

# if __WORDSIZE == 64
    typedef unsigned long int uintptr_t;
# else
    typedef unsigned int uintptr_t;
# endif /* __WORDSIZE == 64 */
# else /* __WORDSIZE */
# if defined(_LP64) || defined(_I32LPx)
    typedef unsigned long int uintptr_t;
# else
    typedef unsigned int uintptr_t;
# endif
# endif /* __WORDSIZE */
# endif /* _WIN32 */

#define _UINTPTR_T
#define _UINTPTR_T_DEFINED
#endif /* !defined(_UINTPTR_T) || !defined(_UINTPTR_T_DEFINED) */

/* Perform an unsigned cast to uintptr_t. */
#define cast_to_pointer_integral_type(value) \
    ((uintptr_t)((size_t)(value)))

/* Perform a cast of a pointer to LargestIntegralType */
#define cast_ptr_to_largest_integral_type(value) \
    cast_to_largest_integral_type(cast_to_pointer_integral_type(value))

/* GCC have printf type attribute check. */
#ifndef __GNUC__
#define CMOCKA_PRINTF_ATTRIBUTE(a,b) \

```

```

__attribute__ ((__format__ (__printf__, a, b)))
#else

#define CMOCKA_PRINTF_ATTRIBUTE(a,b)

#endif /* __GNUC__ */

#if defined(__GNUC__)

#define CMOCKA_DEPRECATED __attribute__ ((deprecated))
#elif defined(_MSC_VER)

#define CMOCKA_DEPRECATED __declspec(deprecated)
#else

#define CMOCKA_DEPRECATED

#endif

#define WILL_RETURN_ALWAYS -1
#define WILL_RETURN_ONCE -2

/***
* @defgroup cmocka_mock Mock Objects
* @ingroup cmocka
*
* Mock objects mock objects are simulated objects that mimic the behavior of
* real objects. Instead of calling the real objects, the tested object calls a
* mock object that merely asserts that the correct methods were called, with
* the expected parameters, in the correct order.
*
* <ul>
* <li><strong>will_return(function, value)</strong> - The will_return() macro
* pushes a value onto a stack of mock values. This macro is intended to be
* used by the unit test itself, while programming the behaviour of the mocked

```

```
* object.</li>
*
* <li><strong>mock()</strong> - the mock macro pops a value from a stack of
* test values. The user of the mock() macro is the mocked object that uses it
* to learn how it should behave.</li>
* </ul>
*
* Because the will_return() and mock() are intended to be used in pairs, the
* cmocka library would fail the test if there are more values pushed onto the
* stack using will_return() than consumed with mock() and vice-versa.
*
* The following unit test stub illustrates how would a unit test instruct the
* mock object to return a particular value:
*
* @code
* will_return(chef_cook, "hotdog");
* will_return(chef_cook, 0);
* @endcode
*
* Now the mock object can check if the parameter it received is the parameter
* which is expected by the test driver. This can be done the following way:
*
* @code
* int chef_cook(const char *order, char **dish_out)
* {
*     check_expected(order);
* }
* @endcode
*
```

```

* For a complete example please at a look

* <a
http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef\_wrap/waiter\_test\_wrap.c>here</a>.

*

* #{@

*/



#ifndef DOXYGEN

/**

* @brief Retrieve a return value of the current function.

*

* @return The value which was stored to return by this function.

*

* @see will_return()

*/

LargestIntegralType mock(void);

#else

#define mock() _mock(__func__, __FILE__, __LINE__)

#endif


#ifndef DOXYGEN

/**

* @brief Retrieve a typed return value of the current function.

*

* The value would be casted to type internally to avoid having the
* caller to do the cast manually.

*

* @param[in] #type The expected type of the return value

```

```
*  
* @return The value which was stored to return by this function.  
*  
* @code  
* int param;  
*  
* param = mock_type(int);  
* @endcode  
*  
* @see will_return()  
* @see mock()  
* @see mock_ptr_type()  
*/  
  
#type mock_type(#type);  
  
#else  
  
#define mock_type(type) ((type) mock())  
  
#endif  
  
  
#ifdef DOXYGEN  
/**  
 * @brief Retrieve a typed return value of the current function.  
 *  
 * The value would be casted to type internally to avoid having the  
 * caller to do the cast manually but also casted to uintptr_t to make  
 * sure the result has a valid size to be used as a pointer.  
 *  
 * @param[in] #type The expected type of the return value  
 *  
 * @return The value which was stored to return by this function.
```

```

*
* @code
* char *param;
*
* param = mock_ptr_type(char *);
*
* @endcode
*
* @see will_return()
* @see mock()
* @see mock_type()
*/
type mock_ptr_type(#type);
#else
#define mock_ptr_type(type) ((type) (uintptr_t) mock())
#endif

```

```

#endif DOXYGEN
/** 
* @brief Store a value to be returned by mock() later.
*
* @param[in] #function The function which should return the given value.
*
* @param[in] value The value to be returned by mock().
*
* @code
* int return_integer(void)
{
*     return (int)mock();
}

```

```

* }

*

* static void test_integer_return(void **state)
{
    will_return(return_integer, 42);

    assert_int_equal(my_function_calling_return_integer(), 42);
}

* @endcode

*

* @see mock()
* @see will_return_count()
*/
void will_return(#function, LargestIntegralType value);

#else

#define will_return(function, value) \
    _will_return(#function, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), 1)

#endif

#ifndef DOXYGEN
/**

* @brief Store a value to be returned by mock() later.
*
* @param[in] #function The function which should return the given value.
*
* @param[in] value The value to be returned by mock().
*
* @param[in] count The parameter indicates the number of times the value should

```

```
*      be returned by mock(). If count is set to -1, the value
*
*      will always be returned but must be returned at least once.
*
*      If count is set to -2, the value will always be returned
*
*          by mock(), but is not required to be returned.
*
*
* @see mock()
*/
void will_return_count(#function, LargestIntegralType value, int count);
#else
#define will_return_count(function, value, count) \
    _will_return(function, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)
#endif
```

```
#ifdef DOXYGEN
/**
 * @brief Store a value that will be always returned by mock().
 *
 * @param[in] #function The function which should return the given value.
 *
 * @param[in] #value The value to be returned by mock().
 *
 * This is equivalent to:
 *
 * @code
 * will_return_count(function, value, -1);
 * @endcode
 *
 * @see will_return_count()
 * @see mock()
```

```
*/  
  
void will_return_always(#function, LargestIntegralType value);  
  
#else  
  
#define will_return_always(function, value) \  
    will_return_count(function, (value), WILL_RETURN_ALWAYS)  
  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Store a value that may be always returned by mock().  
 *  
 * This stores a value which will always be returned by mock() but is not  
 * required to be returned by at least one call to mock(). Therefore,  
 * in contrast to will_return_always() which causes a test failure if it  
 * is not returned at least once, will_return_maybe() will never cause a test  
 * to fail if its value is not returned.  
 *  
 * @param[in] #function The function which should return the given value.  
 *  
 * @param[in] #value The value to be returned by mock().  
 *  
 * This is equivalent to:  
 * @code  
 * will_return_count(function, value, -2);  
 * @endcode  
 *  
 * @see will_return_count()  
 * @see mock()  
*/
```

```
void will_return_maybe(#function, LargestIntegralType value);

#else

#define will_return_maybe(function, value) \
    will_return_count(function, (value), WILL_RETURN_ONCE)

#endif

/** @} */

/*
 * @defgroup cmocka_param Checking Parameters
 * @ingroup cmocka
 *
 * Functionality to store expected values for mock function parameters.
 *
 * In addition to storing the return values of mock functions, cmocka provides
 * functionality to store expected values for mock function parameters using
 * the expect_*() functions provided. A mock function parameter can then be
 * validated using the check_expected() macro.
 *
 * Successive calls to expect_*() macros for a parameter queues values to check
 * the specified parameter. check_expected() checks a function parameter
 * against the next value queued using expect_(), if the parameter check fails
 * a test failure is signalled. In addition if check_expected() is called and
 * no more parameter values are queued a test failure occurs.
 *
 * The following test stub illustrates how to do this. First is the the function
 * we call in the test driver:
 *
 * @code
 * static void test_driver(void **state)
 */
```

```

* {
*   expect_string(chef_cook, order, "hotdog");
* }
* @endcode
*
* Now the chef_cook function can check if the parameter we got passed is the
* parameter which is expected by the test driver. This can be done the
* following way:
*
* @code
* int chef_cook(const char *order, char **dish_out)
* {
*   check_expected(order);
* }
* @endcode
*
* For a complete example please at a look at
* <a href="http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef_wrap/waiter_test_wrap.c">here</a>
*
* @{
*/

```

```

*/
#ifndef DOXYGEN

/**

 * @brief Add a custom parameter checking function.

 *

 * If the event parameter is NULL the event structure is allocated internally

 * by this function. If the parameter is provided it must be allocated on the

 * heap and doesn't need to be deallocated by the caller.

 *

 * @param[in] #function The function to add a custom parameter checking

 *           function for.

 *

 * @param[in] #parameter The parameters passed to the function.

 *

 * @param[in] #check_function The check function to call.

 *

 * @param[in] check_data     The data to pass to the check function.

 */

void expect_check(#function, #parameter, #check_function, const void *check_data);

#else

#define expect_check(function, parameter, check_function, check_data) \
    _expect_check(function, parameter, __FILE__, __LINE__, check_function, \
                  cast_to_largest_integral_type(check_data), NULL, 1)

#endif

#endif DOXYGEN

/**

 * @brief Add an event to check if the parameter value is part of the provided

 *       array.


```

```
*  
* The event is triggered by calling check_expected() in the mocked function.  
*  
* @param[in] #function The function to add the check for.  
*  
* @param[in] #parameter The name of the parameter passed to the function.  
*  
* @param[in] value_array[] The array to check for the value.  
*  
* @see check_expected().  
*/  
  
void expect_in_set(#function, #parameter, LargestIntegralType value_array[]);  
  
#else  
  
#define expect_in_set(function, parameter, value_array) \  
    expect_in_set_count(function, parameter, value_array, 1)  
  
#endif  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Add an event to check if the parameter value is part of the provided  
 *        array.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] value_array[] The array to check for the value.  
 */
```

```

*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);
#else
#define expect_in_set_count(function, parameter, value_array, count) \
    _expect_in_set(function, parameter, __FILE__, __LINE__, value_array, \
        sizeof(value_array) / sizeof((value_array)[0]), count)
#endif

#ifndef DOXYGEN
/***
* @brief Add an event to check if the parameter value is not part of the
*        provided array.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @see check_expected().
*/

```

```

void expect_not_in_set(#function, #parameter, LargestIntegralType value_array[]);

#else

#define expect_not_in_set(function, parameter, value_array) \
    expect_not_in_set_count(function, parameter, value_array, 1)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter value is not part of the
 *        provided array.

 *
 * The event is triggered by calling check_expected() in the mocked function.

 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] value_array[] The array to check for the value.
 *
 * @param[in] count The count parameter returns the number of times the value
 *                 should be returned by check_expected(). If count is set
 *                 to -1 the value will always be returned.
 *
 * @see check_expected().
 */

void expect_not_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);

#else

#define expect_not_in_set_count(function, parameter, value_array, count) \
    _expect_not_in_set( \

```

```

#define DOXYGEN

/* @brief Add an event to check a parameter is inside a numerical range.
 * The check would succeed if minimum <= value <= maximum.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] minimum The lower boundary of the interval to check against.
 *
 * @param[in] maximum The upper boundary of the interval to check against.
 *
 * @see check_expected().
 */

void expect_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum);

#else

#define expect_in_range(function, parameter, minimum, maximum) \
expect_in_range_count(function, parameter, minimum, maximum, 1)

#endif

```

```
#ifdef DOXYGEN

/**

 * @brief Add an event to repeatedly check a parameter is inside a
 * numerical range. The check would succeed if minimum <= value <= maximum.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] minimum The lower boundary of the interval to check against.
 *
 * @param[in] maximum The upper boundary of the interval to check against.
 *
 * @param[in] count The count parameter returns the number of times the value
 * should be returned by check_expected(). If count is set
 * to -1 the value will always be returned.
 *
 * @see check_expected().
 */

void expect_in_range_count(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum, size_t count);

#else

#define expect_in_range_count(function, parameter, minimum, maximum, count) \
_expect_in_range(function, parameter, __FILE__, __LINE__, minimum, \
maximum, count)

#endif
```

```

#ifndef DOXYGEN

/**

 * @brief Add an event to check a parameter is outside a numerical range.
 *
 * The check would succeed if minimum > value > maximum.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] minimum The lower boundary of the interval to check against.
 *
 * @param[in] maximum The upper boundary of the interval to check against.
 *
 * @see check_expected().
 */

void expect_not_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum);

#else

#define expect_not_in_range(function, parameter, minimum, maximum) \
    expect_not_in_range_count(function, parameter, minimum, maximum, 1)

#endif

```

```

#ifndef DOXYGEN

/**

 * @brief Add an event to repeatedly check a parameter is outside a
 * numerical range. The check would succeed if minimum > value > maximum.
 *

```

```

* The event is triggered by calling check_expected() in the mocked function.

*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/

```

```
void expect_not_in_range_count(#function, #parameter, LargestIntegralType minimum,
LargestIntegralType maximum, size_t count);
```

```
#else
```

```
#define expect_not_in_range_count(function, parameter, minimum, maximum, \
                                count) \
    _expect_not_in_range(function, parameter, __FILE__, __LINE__, \
                         minimum, maximum, count)
```

```
#endif
```

```
#ifdef DOXYGEN
```

```
/**
```

```
* @brief Add an event to check if a parameter is the given value.
*
```

```
* The event is triggered by calling check_expected() in the mocked function.  
*  
* @param[in] #function The function to add the check for.  
*  
* @param[in] #parameter The name of the parameter passed to the function.  
*  
* @param[in] value The value to check.  
*  
* @see check_expected().  
*/  
  
void expect_value(#function, #parameter, LargestIntegralType value);  
  
#else  
  
#define expect_value(function, parameter, value) \  
    expect_value_count(function, parameter, value, 1)  
  
#endif  
  
  
#ifdef DOXYGEN  
  
/**  
 * @brief Add an event to repeatedly check if a parameter is the given value.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] value The value to check.  
 *  
 * @param[in] count The count parameter returns the number of times the value
```

```

*      should be returned by check_expected(). If count is set
*      to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_value_count(#function, #parameter, LargestIntegralType value, size_t count);
#else
#define expect_value_count(function, parameter, value, count) \
    _expect_value(function, parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)
#endif

#ifndef DOXYGEN
/** 
 * @brief Add an event to check if a parameter isn't the given value.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] value The value to check.
 *
 * @see check_expected().
*/
void expect_not_value(#function, #parameter, LargestIntegralType value);
#else
#define expect_not_value(function, parameter, value) \

```

```

expect_not_value_count(function, parameter, value, 1)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to repeatedly check if a parameter isn't the given value.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *

 * @param[in] #parameter The name of the parameter passed to the function.

 *

 * @param[in] value The value to check.

 *

 * @param[in] count The count parameter returns the number of times the value

 *                 should be returned by check_expected(). If count is set

 *                 to -1 the value will always be returned.

 *

 * @see check_expected().

 */

void expect_not_value_count(#function, #parameter, LargestIntegralType value, size_t count);

#else

#define expect_not_value_count(function, parameter, value, count) \
    _expect_not_value(#function, #parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)

#endif

#ifndef DOXYGEN

```

```
/**  
 * @brief Add an event to check if the parameter value is equal to the  
 *        provided string.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *  
 * @param[in] #parameter The name of the parameter passed to the function.  
 *  
 * @param[in] string The string value to compare.  
 *  
 * @see check_expected().  
 */  
  
void expect_string(#function, #parameter, const char *string);  
  
#else  
  
#define expect_string(function, parameter, string) \  
    expect_string_count(function, parameter, string, 1)  
  
#endif  
  
#ifdef DOXYGEN  
/**  
 * @brief Add an event to check if the parameter value is equal to the  
 *        provided string.  
 *  
 * The event is triggered by calling check_expected() in the mocked function.  
 *  
 * @param[in] #function The function to add the check for.  
 *
```

```

* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_string_count(#function, #parameter, const char *string, size_t count);
#else
#define expect_string_count(function, parameter, string, count) \
    _expect_string(#function, #parameter, __FILE__, __LINE__, \
                  (const char*)(string), count)
#endif

#ifndef DOXYGEN
/**
* @brief Add an event to check if the parameter value isn't equal to the
*        provided string.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.

```

```

*
* @see check_expected().
*/
void expect_not_string(#function, #parameter, const char *string);
#else
#define expect_not_string(function, parameter, string) \
    expect_not_string_count(function, parameter, string, 1)
#endif

#ifndef DOXYGEN
/***
* @brief Add an event to check if the parameter value isn't equal to the
*       provided string.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                 should be returned by check_expected(). If count is set
*                 to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_string_count(#function, #parameter, const char *string, size_t count);

```

```

#else

#define expect_not_string_count(function, parameter, string, count) \
    _expect_not_string(#function, #parameter, __FILE__, __LINE__, \
        (const char*)(string), count)

#endif

#ifndef DOXYGEN

/**

 * @brief Add an event to check if the parameter does match an area of memory.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *

 * @param[in] #parameter The name of the parameter passed to the function.

 *

 * @param[in] memory The memory to compare.

 *

 * @param[in] size The size of the memory to compare.

 *

 * @see check_expected().

 */

void expect_memory(#function, #parameter, void *memory, size_t size);

#else

#define expect_memory(function, parameter, memory, size) \
    expect_memory_count(function, parameter, memory, size, 1)

#endif

#ifndef DOXYGEN

```

```

/**

 * @brief Add an event to repeatedly check if the parameter does match an area
 *        of memory.

 *

 * The event is triggered by calling check_expected() in the mocked function.

 *

 * @param[in] #function The function to add the check for.

 *

 * @param[in] #parameter The name of the parameter passed to the function.

 *

 * @param[in] memory The memory to compare.

 *

 * @param[in] size The size of the memory to compare.

 *

 * @param[in] count The count parameter returns the number of times the value
 *        should be returned by check_expected(). If count is set
 *        to -1 the value will always be returned.

 *

 * @see check_expected().

 */

void expect_memory_count(#function, #parameter, void *memory, size_t size, size_t count);

#else

#define expect_memory_count(function, parameter, memory, size, count) \
    _expect_memory(#function, #parameter, __FILE__, __LINE__, \
                  (const void*)(memory), size, count)

#endif

#ifndef DOXYGEN

/**
```

```
* @brief Add an event to check if the parameter doesn't match an area of
*      memory.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] memory The memory to compare.
*
* @param[in] size The size of the memory to compare.
*
* @see check_expected().
*/
void expect_not_memory(#function, #parameter, void *memory, size_t size);
#else
#define expect_not_memory(function, parameter, memory, size) \
    expect_not_memory_count(function, parameter, memory, size, 1)
#endif

#ifndef DOXYGEN
/** 
* @brief Add an event to repeatedly check if the parameter doesn't match an
*      area of memory.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
```

```

*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] memory The memory to compare.
*
* @param[in] size The size of the memory to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_memory_count(#function, #parameter, void *memory, size_t size, size_t count);
#else
#define expect_not_memory_count(function, parameter, memory, size, count) \
    _expect_not_memory(#function, #parameter, __FILE__, __LINE__, \
                      (const void*)(memory), size, count)
#endif

#ifndef DOXYGEN
/**
* @brief Add an event to check if a parameter (of any value) has been passed.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*

```

```

* @param[in] #parameter The name of the parameter passed to the function.
*
* @see check_expected().
*/
void expect_any(#function, #parameter);
#else
#define expect_any(function, parameter) \
    expect_any_count(function, parameter, 1)
#endif

#ifndef DOXYGEN
/***
* @brief Add an event to repeatedly check if a parameter (of any value) has
* been passed.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] count The count parameter returns the number of times the value
* should be returned by check_expected(). If count is set
* to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_any_count(#function, #parameter, size_t count);
#else

```

```

#define expect_any_count(function, parameter, count) \
    _expect_any(#function, #parameter, __FILE__, __LINE__, count)

#endif

#ifndef DOXYGEN
/**

 * @brief Determine whether a function parameter is correct.

 *
 * This ensures the next value queued by one of the expect_*() macros matches
 * the specified variable.

 *
 * This function needs to be called in the mock object.

 *
 * @param[in] #parameter The parameter to check.

 */
void check_expected(#parameter);

#else

#define check_expected(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
                    cast_to_largest_integral_type(parameter))

#endif

#ifndef DOXYGEN
/**

 * @brief Determine whether a function parameter is correct.

 *
 * This ensures the next value queued by one of the expect_*() macros matches
 * the specified variable.

 *

```

```

* This function needs to be called in the mock object.

*
* @param[in] #parameter The pointer to check.
*/
void check_expected_ptr(#parameter);

#else

#define check_expected_ptr(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
                    cast_ptr_to_largest_integral_type(parameter))

#endif

/** @} */

/**
* @defgroup cmocka_asserts Assert Macros
* @ingroup cmocka
*
* This is a set of useful assert macros like the standard C library's
* assert(3) macro.
*
* On an assertion failure a cmocka assert macro will write the failure to the
* standard error stream and signal a test failure. Due to limitations of the C
* language the general C standard library assert() and cmocka's assert_true()
* and assert_false() macros can only display the expression that caused the
* assert failure. cmocka's type specific assert macros, assert_{type}_equal()
* and assert_{type}_not_equal(), display the data that caused the assertion
* failure which increases data visibility aiding debugging of failing test
* cases.
*

```

```
* @@
*/
#ifndef DOXYGEN
/**

* @brief Assert that the given expression is true.

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if expression is false (i.e., compares equal to
* zero).

*
* @param[in] expression The expression to evaluate.

*
* @see assert_int_equal()
* @see assert_string_equal()
*/
void assert_true(scalar expression);

#else

#define assert_true(c) _assert_true(cast_to_largest_integral_type(c), #c, \
                           __FILE__, __LINE__)

#endif

#ifndef DOXYGEN
/**

* @brief Assert that the given expression is false.

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if expression is true.

*
```

```

* @param[in] expression The expression to evaluate.
*
* @see assert_int_equal()
* @see assert_string_equal()
*/
void assert_false(scalar expression);

#else

#define assert_false(c) _assert_true(!(cast_to_largest_integral_type(c)), #c, \
                                         __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Assert that the return_code is greater than or equal to 0.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the return code is smaller than 0. If the function
* you check sets an errno if it fails you can pass it to the function and
* it will be printed as part of the error message.
*
* @param[in] rc      The return code to evaluate.
*
* @param[in] error   Pass errno here or 0.
*/
void assert_return_code(int rc, int error);

#else

#define assert_return_code(rc, error) \
    _assert_return_code(cast_to_largest_integral_type(rc), \
                        sizeof(rc), \

```

```
    cast_to_largest_integral_type(error), \
    #rc, __FILE__, __LINE__)
#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the given pointer is non-NULL.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the pointer is non-NULL.

 *

 * @param[in] pointer The pointer to evaluate.

 *

 * @see assert_null()

 */

void assert_non_null(void *pointer);

#else

#define assert_non_null(c) _assert_true(cast_ptr_to_largest_integral_type(c), #c, \
                                         __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the given pointer is NULL.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the pointer is non-NULL.

 *

 * @param[in] pointer The pointer to evaluate.

 *
```

```

*
* @see assert_non_null()
*/
void assert_null(void *pointer);

#else

#define assert_null(c) _assert_true(!(cast_ptr_to_largest_integral_type(c)), #c, \
__FILE__, __LINE__)
#endif

#endif DOXYGEN

/***
* @brief Assert that the two given pointers are equal.
*
* The function prints an error message and terminates the test by calling
* fail() if the pointers are not equal.
*
* @param[in] a      The first pointer to compare.
*
* @param[in] b      The pointer to compare against the first one.
*/
void assert_ptr_equal(void *a, void *b);

#else

#define assert_ptr_equal(a, b) \
_assert_int_equal(cast_ptr_to_largest_integral_type(a), \
cast_ptr_to_largest_integral_type(b), \
__FILE__, __LINE__)

#endif

#endif DOXYGEN

```

```

/**

 * @brief Assert that the two given pointers are not equal.

 *

 * The function prints an error message and terminates the test by calling

 * fail() if the pointers are equal.

 *

 * @param[in] a      The first pointer to compare.

 *

 * @param[in] b      The pointer to compare against the first one.

 */

void assert_ptr_not_equal(void *a, void *b);

#else

#define assert_ptr_not_equal(a, b) \
    _assert_int_not_equal(cast_ptr_to_largest_integral_type(a), \
                          cast_ptr_to_largest_integral_type(b), \
                          __FILE__, __LINE__)

#endif

#endif DOXYGEN

/**

 * @brief Assert that the two given integers are equal.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the integers are not equal.

 *

 * @param[in] a  The first integer to compare.

 *

 * @param[in] b  The integer to compare against the first one.

*/

```

```
void assert_int_equal(int a, int b);

#else

#define assert_int_equal(a, b) \
    _assert_int_equal(cast_to_largest_integral_type(a), \
                      cast_to_largest_integral_type(b), \
                      __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/** 
 * @brief Assert that the two given integers are not equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the integers are equal.
 *
 * @param[in] a The first integer to compare.
 *
 * @param[in] b The integer to compare against the first one.
 *
 * @see assert_int_equal()
 */

void assert_int_not_equal(int a, int b);

#else

#define assert_int_not_equal(a, b) \
    _assert_int_not_equal(cast_to_largest_integral_type(a), \
                          cast_to_largest_integral_type(b), \
                          __FILE__, __LINE__)

#endif
```

```
#ifdef DOXYGEN

/**

 * @brief Assert that the two given strings are equal.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the strings are not equal.

 *

 * @param[in] a The string to check.

 *

 * @param[in] b The other string to compare.

 */

void assert_string_equal(const char *a, const char *b);

#else

#define assert_string_equal(a, b) \

    _assert_string_equal((const char*)(a), (const char*)(b), __FILE__, \

        __LINE__)

#endif

#endif DOXYGEN

/**

 * @brief Assert that the two given strings are not equal.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the strings are equal.

 *

 * @param[in] a The string to check.

 *

 * @param[in] b The other string to compare.

*/
```

```
void assert_string_not_equal(const char *a, const char *b);

#else

#define assert_string_not_equal(a, b) \
    _assert_string_not_equal((const char*)(a), (const char*)(b), __FILE__, \
                           __LINE__)

#endif

#ifndef DOXYGEN

/**

 * @brief Assert that the two given areas of memory are equal, otherwise fail.

 *

 * The function prints an error message to standard error and terminates the

 * test by calling fail() if the memory is not equal.

 *

 * @param[in] a The first memory area to compare

 *           (interpreted as unsigned char).

 *

 * @param[in] b The second memory area to compare

 *           (interpreted as unsigned char).

 *

 * @param[in] size The first n bytes of the memory areas to compare.

 */

void assert_memory_equal(const void *a, const void *b, size_t size);

#else

#define assert_memory_equal(a, b, size) \
    _assert_memory_equal((const void*)(a), (const void*)(b), size, __FILE__, \
                         __LINE__)

#endif
```

```
#ifdef DOXYGEN
<太后
 * @brief Assert that the two given areas of memory are not equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the memory is equal.
 *
 * @param[in] a The first memory area to compare
 *             (interpreted as unsigned char).
 *
 * @param[in] b The second memory area to compare
 *             (interpreted as unsigned char).
 *
 * @param[in] size The first n bytes of the memory areas to compare.
 */

```

```
void assert_memory_not_equal(const void *a, const void *b, size_t size);
```

```
#else
```

```
#define assert_memory_not_equal(a, b, size) \
    _assert_memory_not_equal((const void*)(a), (const void*)(b), size, \
                            __FILE__, __LINE__)
```

```
#endif
```

```
#ifdef DOXYGEN
```

```
/***
 * @brief Assert that the specified value is not smaller than the minimum
 * and not greater than the maximum.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if value is not in range.
```

```

*
* @param[in] value The value to check.
*
* @param[in] minimum The minimum value allowed.
*
* @param[in] maximum The maximum value allowed.
*/
void assert_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType
maximum);

#else

#define assert_in_range(value, minimum, maximum) \
    _assert_in_range( \
        cast_to_largest_integral_type(value), \
        cast_to_largest_integral_type(minimum), \
        cast_to_largest_integral_type(maximum), __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Assert that the specified value is smaller than the minimum or
* greater than the maximum.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is in range.
*
* @param[in] value The value to check.
*
* @param[in] minimum The minimum value to compare.
*

```

```

* @param[in] maximum The maximum value to compare.

*/
void assert_not_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType
maximum);

#else

#define assert_not_in_range(value, minimum, maximum) \
    _assert_not_in_range( \
        cast_to_largest_integral_type(value), \
        cast_to_largest_integral_type(minimum), \
        cast_to_largest_integral_type(maximum), __FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Assert that the specified value is within a set.

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is not within a set.

*
* @param[in] value The value to look up
*
* @param[in] values[] The array to check for the value.
*
* @param[in] count The size of the values array.

*/
void assert_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);

#else

#define assert_in_set(value, values, number_of_values) \
    _assert_in_set(value, values, number_of_values, __FILE__, __LINE__)


```

```
#endif

#ifndef DOXYGEN

$$\begin{aligned} \text{* @brief Assert that the specified value is not within a set.} \\ \text{*} \\ \text{* The function prints an error message to standard error and terminates the} \\ \text{* test by calling fail() if value is within a set.} \\ \text{*} \\ \text{* @param[in] value The value to look up} \\ \text{*} \\ \text{* @param[in] values[] The array to check for the value.} \\ \text{*} \\ \text{* @param[in] count The size of the values array.} \\ \end{aligned}$$

*/
void assert_not_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);

#else
#define assert_not_in_set(value, values, number_of_values) \
    _assert_not_in_set(value, values, number_of_values, __FILE__, __LINE__)
#endif

/** @} */


$$\begin{aligned} \text{* @defgroup cmocka_call_order Call Ordering} \\ \text{* @ingroup cmocka} \\ \text{*} \\ \text{* It is often beneficial to make sure that functions are called in an} \\ \text{* order. This is independent of mock returns and parameter checking as both} \end{aligned}$$

```

- * of the aforementioned do not check the order in which they are called from different functions.
- *
- *
- * expect_function_call(function) - The
- * expect_function_call() macro pushes an expectation onto the stack of
- * expected calls.
- *
- * function_called() - pops a value from the stack of
- * expected calls. function_called() is invoked within the mock object
- * that uses it.
- *
- *
- * expect_function_call() and function_called() are intended to be used in
- * pairs. Cmocka will fail a test if there are more or less expected calls
- * created (e.g. expect_function_call()) than consumed with function_called().
- * There are provisions such as ignore_function_calls() which allow this
- * restriction to be circumvented in tests where mock calls for the code under
- * test are not the focus of the test.
- *
- * The following example illustrates how a unit test instructs cmocka
- * to expect a function_called() from a particular mock,
- * chef_sing():
- *
- * @code
- * void chef_sing(void);
- *
- * void code_under_test()
- * {

```
* chef_sing();  
* }  
*  
* void some_test(void **state)  
* {  
*   expect_function_call(chef_sing);  
*   code_under_test();  
* }  
* @endcode  
*  
* The implementation of the mock then must check whether it was meant to  
* be called by invoking <strong>function_called()</strong>:  
*  
* @code  
* void chef_sing()  
* {  
*   function_called();  
* }  
* @endcode  
*  
* @{  
*/
```

```
#ifdef DOXYGEN  
/**  
* @brief Check that current mocked function is being called in the expected  
*       order  
*  
* @see expect_function_call()
```

```

*/
void function_called(void);
#else
#define function_called() _function_called(__func__, __FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/**

 * @brief Store expected call(s) to a mock to be checked by function_called()
 *
 *      later.
 *
 * @param[in] #function The function which should should be called
 *
 * @param[in] times number of times this mock must be called
 *
 * @see function_called()
*/
void expect_function_calls(#function, const int times);
#else
#define expect_function_calls(function, times) \
    _expect_function_call(#function, __FILE__, __LINE__, times)
#endif

#ifndef DOXYGEN
/**

 * @brief Store expected single call to a mock to be checked by
 *      function_called() later.
 *
 * @param[in] #function The function which should should be called

```

```

*
* @see function_called()
*/
void expect_function_call(#function);
#else
#define expect_function_call(function) \
_expect_function_call(function, __FILE__, __LINE__, 1)
#endif

#ifndef DOXYGEN
/***
* @brief Expects function_called() from given mock at least once
*
* @param[in] #function The function which should be called
*
* @see function_called()
*/
void expect_function_call_any(#function);
#else
#define expect_function_call_any(function) \
_expect_function_call(function, __FILE__, __LINE__, -1)
#endif

#ifndef DOXYGEN
/***
* @brief Ignores function_called() invocations from given mock function.
*
* @param[in] #function The function which should be called
*

```

```
* @see function_called()  
*/  
  
void ignore_function_calls(#function);  
  
#else  
  
#define ignore_function_calls(function) \  
    _expect_function_call(#function, __FILE__, __LINE__, -2)  
  
#endif  
  
/** @} */  
  
/**  
 * @defgroup cmocka_exec Running Tests  
 * @ingroup cmocka  
 *  
 * This is the way tests are executed with CMocka.  
 *  
 * The following example illustrates this macro's use with the unit_test macro.  
 *  
 * @code  
 * void Test0(void **state);  
 * void Test1(void **state);  
 *  
 * int main(void)  
 * {  
 *     const struct CMUnitTest tests[] = {  
 *         cmocka_unit_test(Test0),  
 *         cmocka_unit_test(Test1),  
 *     };  
 *  
 *
```

```
*     return cmocka_run_group_tests(tests, NULL, NULL);

* }

* @endcode

*

* @{
* /
#endif DOXYGEN

/**

* @brief Forces the test to fail immediately and quit.

*/
void fail(void);

#else

#define fail() _fail(__FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Forces the test to not be executed, but marked as skipped

*/
void skip(void);

#else

#define skip() _skip(__FILE__, __LINE__)

#endif

#ifndef DOXYGEN

/**

* @brief Forces the test to fail immediately and quit, printing the reason.

*

```

```
* @code
* fail_msg("This is some error message for test");

* @endcode
*
* or
*
* @code
* char *error_msg = "This is some error message for test";
* fail_msg("%s", error_msg);
* @endcode
*/
void fail_msg(const char *msg, ...);

#else
#define fail_msg(msg, ...) do { \
    print_error("ERROR: " msg "\n", ##__VA_ARGS__); \
    fail(); \
} while (0)

#endif

#endif DOXYGEN
/**

* @brief Generic method to run a single test.
*
* @deprecated This function was deprecated in favor of cmocka_run_group_tests
*
* @param[in] #function The function to test.
*
* @return 0 on success, 1 if an error occurred.
*
```

```

* @code
* // A test case that does nothing and succeeds.

* void null_test_success(void **state) {
*
* }

* int main(void) {
*     return run_test(null_test_success);
*
* }
* @endcode
*/
int run_test(#function);

#else
#define run_test(f) _run_test(#f, f, NULL, UNIT_TEST_FUNCTION_TYPE_TEST, NULL)
#endif

static inline void _unit_test_dummy(void **state) {
    (void)state;
}

/** Initializes a UnitTest structure.
*
* @deprecated This function was deprecated in favor of cmocka_unit_test
*/
#define unit_test(f) { #f, f, UNIT_TEST_FUNCTION_TYPE_TEST }

#define _unit_test_setup(test, setup) \
{ #test " " #setup, setup, UNIT_TEST_FUNCTION_TYPE_SETUP }

/** Initializes a UnitTest structure with a setup function.

```

```
* @deprecated This function was deprecated in favor of cmocka_unit_test_setup
*/
#define unit_test_setup(test, setup) \
    _unit_test_setup(test, setup), \
    unit_test(test), \
    _unit_test_teardown(test, _unit_test_dummy)

#define _unit_test_teardown(test, teardown) \
{ #test "_" #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_TEARDOWN }

/** Initializes a UnitTest structure with a teardown function.
*
* @deprecated This function was deprecated in favor of cmocka_unit_test_teardown
*/
#define unit_test_teardown(test, teardown) \
    _unit_test_setup(test, _unit_test_dummy), \
    unit_test(test), \
    _unit_test_teardown(test, teardown)

/** Initializes a UnitTest structure for a group setup function.
*
* @deprecated This function was deprecated in favor of cmocka_run_group_tests
*/
#define group_test_setup(setup) \
{ "group_" #setup, setup, UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP }

/** Initializes a UnitTest structure for a group teardown function.
*

```

```
* @deprecated This function was deprecated in favor of cmocka_run_group_tests
*/
#define group_test_teardown(teardown) \
{ "group_" #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN }

/** 
 * Initialize an array of UnitTest structures with a setup function for a test
 * and a teardown function. Either setup or teardown can be NULL.
 *
 * @deprecated This function was deprecated in favor of
 * cmocka_unit_test_setup_teardown
*/
#define unit_test_setup_teardown(test, setup, teardown) \
    _unit_test_setup(test, setup), \
    unit_test(test), \
    _unit_test_teardown(test, teardown)

/** Initializes a CMUnitTest structure.*/
#define cmocka_unit_test(f) { #f, f, NULL, NULL, NULL }

/** Initializes a CMUnitTest structure with a setup function.*/
#define cmocka_unit_test_setup(f, setup) { #f, f, setup, NULL, NULL }

/** Initializes a CMUnitTest structure with a teardown function.*/
#define cmocka_unit_test_teardown(f, teardown) { #f, f, NULL, teardown, NULL }

/** 
 * Initialize an array of CMUnitTest structures with a setup function for a test

```

```

* and a teardown function. Either setup or teardown can be NULL.

*/
#define cmocka_unit_test_setup_teardown(f, setup, teardown) { #f, f, setup, teardown, NULL }

/** 
 * Initialize a CMUnitTest structure with given initial state. It will be passed
 * to test function as an argument later. It can be used when test state does
 * not need special initialization or was initialized already.
 * @note If the group setup function initialized the state already, it won't be
 * overridden by the initial state defined here.
*/
#define cmocka_unit_test_prestate(f, state) { #f, f, NULL, NULL, state }

/** 
 * Initialize a CMUnitTest structure with given initial state, setup and
 * teardown function. Any of these values can be NULL. Initial state is passed
 * later to setup function, or directly to test if none was given.
 * @note If the group setup function initialized the state already, it won't be
 * overridden by the initial state defined here.
*/
#define cmocka_unit_test_prestate_setup_teardown(f, setup, teardown, state) { #f, f, setup, teardown,
state }

#define run_tests(tests) _run_tests(tests, sizeof(tests) / sizeof(tests)[0])
#define run_group_tests(tests) _run_group_tests(tests, sizeof(tests) / sizeof(tests)[0])

#ifndef DOXYGEN
/** 
 * @brief Run tests specified by an array of CMUnitTest structures.

```

```
*  
* @param[in] group_tests[] The array of unit tests to execute.  
*  
* @param[in] group_setup The setup function which should be called before  
*           all unit tests are executed.  
*  
* @param[in] group_teardown The teardown function to be called after all  
*           tests have finished.  
*  
* @return 0 on success, or the number of failed tests.  
*  
* @code  
* static int setup(void **state) {  
*     int *answer = malloc(sizeof(int));  
*     if (*answer == NULL) {  
*         return -1;  
*     }  
*     *answer = 42;  
*  
*     *state = answer;  
*  
*     return 0;  
* }  
*  
* static int teardown(void **state) {  
*     free(*state);  
*  
*     return 0;  
* }
```

```
*  
  
* static void null_test_success(void **state) {  
*     (void) state;  
* }  
  
*  
  
* static void int_test_success(void **state) {  
*     int *answer = *state;  
*     assert_int_equal(*answer, 42);  
* }  
  
*  
  
* int main(void) {  
*     const struct CMUnitTest tests[] = {  
*         cmocka_unit_test(null_test_success),  
*         cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),  
*     };  
*  
*     return cmocka_run_group_tests(tests, NULL, NULL);  
* }  
  
* @endcode  
  
*  
  
* @see cmocka_unit_test  
* @see cmocka_unit_test_setup  
* @see cmocka_unit_test_teardown  
* @see cmocka_unit_test_setup_teardown  
*/  
  
int cmocka_run_group_tests(const struct CMUnitTest group_tests[],  
                           CMFixtureFunction group_setup,  
                           CMFixtureFunction group_teardown);  
  
#else
```

```
# define cmocka_run_group_tests(group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(#group_tests, group_tests, sizeof(group_tests) / sizeof(group_tests)[0],  
group_setup, group_teardown)  
#endif  
  
#ifdef DOXYGEN  
/**  
 * @brief Run tests specified by an array of CMUnitTest structures and specify  
 *        a name.  
 *  
 * @param[in] group_name    The name of the group test.  
 *  
 * @param[in] group_tests[] The array of unit tests to execute.  
 *  
 * @param[in] group_setup   The setup function which should be called before  
 *                        all unit tests are executed.  
 *  
 * @param[in] group_teardown The teardown function to be called after all  
 *                          tests have finished.  
 *  
 * @return 0 on success, or the number of failed tests.  
 *  
 * @code  
 * static int setup(void **state) {  
 *     int *answer = malloc(sizeof(int));  
 *     if (*answer == NULL) {  
 *         return -1;  
 *     }  
 *     *answer = 42;  
 *
```

```
*  
  
*     *state = answer;  
  
*  
  
*     return 0;  
  
* }  
  
*  
  
* static int teardown(void **state) {  
  
*     free(*state);  
  
*  
  
*     return 0;  
  
* }  
  
*  
  
* static void null_test_success(void **state) {  
  
*     (void) state;  
  
* }  
  
*  
  
* static void int_test_success(void **state) {  
  
*     int *answer = *state;  
  
*     assert_int_equal(*answer, 42);  
  
* }  
  
*  
  
* int main(void) {  
  
*     const struct CMUnitTest tests[] = {  
  
*         cmocka_unit_test(null_test_success),  
  
*         cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),  
  
*     };  
  
*  
  
*     return cmocka_run_group_tests_name("success_test", tests, NULL, NULL);  
* }
```

```

* @endcode
*
* @see cmocka_unit_test
* @see cmocka_unit_test_setup
* @see cmocka_unit_test_teardown
* @see cmocka_unit_test_setup_teardown
*/
int cmocka_run_group_tests_name(const char *group_name,
                                const struct CMUnitTest group_tests[],
                                CMFixtureFunction group_setup,
                                CMFixtureFunction group_teardown);

#else
#define cmocka_run_group_tests_name(group_name, group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(group_name, group_tests, sizeof(group_tests) / sizeof(group_tests)[0], \
                           group_setup, group_teardown)
#endif

/** @} */

/**
* @defgroup cmocka_alloc Dynamic Memory Allocation
* @ingroup cmocka
*
* Memory leaks, buffer overflows and underflows can be checked using cmocka.
*
* To test for memory leaks, buffer overflows and underflows a module being
* tested by cmocka should replace calls to malloc(), calloc() and free() to
* test_malloc(), test_calloc() and test_free() respectively. Each time a block
* is deallocated using test_free() it is checked for corruption, if a corrupt

```

```
* block is found a test failure is signalled. All blocks allocated using the
* test_*() allocation functions are tracked by the cmocka library. When a test
* completes if any allocated blocks (memory leaks) remain they are reported
* and a test failure is signalled.

*
* For simplicity cmocka currently executes all tests in one process. Therefore
* all test cases in a test application share a single address space which
* means memory corruption from a single test case could potentially cause the
* test application to exit prematurely.

*
* @{
*/

```

```
#ifdef DOXYGEN
/**

* @brief Test function overriding malloc.

*
* @param[in] size The bytes which should be allocated.

*
* @return A pointer to the allocated memory or NULL on error.

*
* @code
* #ifdef UNIT_TESTING
* extern void* _test_malloc(const size_t size, const char* file, const int line);
*
* #define malloc(size) _test_malloc(size, __FILE__, __LINE__)
*
* #endif
*
* void leak_memory() {
```

```

*   int * const temporary = (int*)malloc(sizeof(int));
*   *temporary = 0;
* }
* @endcode
*
* @see malloc(3)
*/
void *test_malloc(size_t size);
#else
#define test_malloc(size) _test_malloc(size, __FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/** 
* @brief Test function overriding calloc.
*
* The memory is set to zero.
*
* @param[in] nmemb The number of elements for an array to be allocated.
*
* @param[in] size The size in bytes of each array element to allocate.
*
* @return A pointer to the allocated memory, NULL on error.
*
* @see calloc(3)
*/
void *test_calloc(size_t nmemb, size_t size);
#else
#define test_calloc(num, size) _test_calloc(num, size, __FILE__, __LINE__)

```

```

#endif

#ifndef DOXYGEN

$$\begin{aligned} \text{/**} \\ * @brief Test function overriding realloc which detects buffer overruns \\ * and memoery leaks. \\ * \\ * @param[in] ptr The memory block which should be changed. \\ * \\ * @param[in] size The bytes which should be allocated. \\ * \\ * @return The newly allocated memory block, NULL on error. \\ */ \end{aligned}$$

void *test_realloc(void *ptr, size_t size);

#else
#define test_realloc(ptr, size) _test_realloc(ptr, size, __FILE__, __LINE__)
#endif

#ifndef DOXYGEN

$$\begin{aligned} \text{/**} \\ * @brief Test function overriding free(3). \\ * \\ * @param[in] ptr The pointer to the memory space to free. \\ * \\ * @see free(3). \\ */ \end{aligned}$$

void test_free(void *ptr);

#else
#define test_free(ptr) _test_free(ptr, __FILE__, __LINE__)
\end{pre}

```

```
#endif

/* Redirect malloc, calloc and free to the unit test allocators. */

#ifndef UNIT_TESTING

#define malloc test_malloc
#define realloc test_realloc
#define calloc test_calloc
#define free test_free

#endif /* UNIT_TESTING */

/** @} */
```

```
/**

 * @defgroup cmocka_mock_assert Standard Assertions
 * @ingroup cmocka
 *
 * How to handle assert(3) of the standard C library.
 *
 * Runtime assert macros like the standard C library's assert() should be
 * redefined in modules being tested to use cmocka's mock_assert() function.
 *
 * Normally mock_assert() signals a test failure. If a function is called using
 * the expect_assert_failure() macro, any calls to mock_assert() within the
 * function will result in the execution of the test. If no calls to
 * mock_assert() occur during the function called via expect_assert_failure() a
 * test failure is signalled.
 *
 * @{
 */
```

```
/**  
 * @brief Function to replace assert(3) in tested code.  
 *  
 * In conjunction with check_assert() it's possible to determine whether an  
 * assert condition has failed without stopping a test.  
 *  
 * @param[in] result The expression to assert.  
 *  
 * @param[in] expression The expression as string.  
 *  
 * @param[in] file The file mock_assert() is called.  
 *  
 * @param[in] line The line mock_assert() is called.  
 *  
 * @code  
 * #ifdef UNIT_TESTING  
 * extern void mock_assert(const int result, const char* const expression,  
 *                         const char * const file, const int line);  
 *  
 * #undef assert  
 * #define assert(expression) \  
 *     mock_assert((int)(expression), #expression, __FILE__, __LINE__);  
 * #endif  
 *  
 * void increment_value(int * const value) {  
 *     assert(value);  
 *     (*value)++;  
 * }
```

```
* @endcode
*
* @see assert(3)
* @see expect_assert_failure
*/
void mock_assert(const int result, const char* const expression,
                 const char * const file, const int line);

#ifndef DOXYGEN
/**

* @brief Ensure that mock_assert() is called.
*
* If mock_assert() is called the assert expression string is returned.
*
* @param[in] fn_call The function will will call mock_assert().
*
* @code
* #define assert mock_assert
*
* void showmessage(const char *message) {
*   assert(message);
* }
*
* int main(int argc, const char* argv[]) {
*   expect_assert_failure(show_message(NULL));
*   printf("succeeded\n");
*   return 0;
* }
* @endcode
*/
```

```

/*
*/
void expect_assert_failure(function fn_call);

#else
#define expect_assert_failure(function_call) \
{ \
    const int result = setjmp(global_expect_assert_env); \
    global_expecting_assert = 1; \
    if (result) { \
        print_message("Expected assertion %s occurred\n", \
                     global_last_failed_assert); \
        global_expecting_assert = 0; \
    } else { \
        function_call; \
        global_expecting_assert = 0; \
        print_error("Expected assert in %s\n", #function_call); \
        _fail(__FILE__, __LINE__); \
    } \
}
#endif

/** @} */

/* Function prototype for setup, test and teardown functions. */
typedef void (*UnitTestFunction)(void **state);

/* Function that determines whether a function parameter value is correct. */
typedef int (*CheckParameterValue)(const LargestIntegralType value,
                                  const LargestIntegralType check_value_data);

```

```
/* Type of the unit test function. */
typedef enum UnitTestFunctionType {
    UNIT_TEST_FUNCTION_TYPE_TEST = 0,
    UNIT_TEST_FUNCTION_TYPE_SETUP,
    UNIT_TEST_FUNCTION_TYPE_TEARDOWN,
    UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP,
    UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN,
} UnitTestFunctionType;

/*
 * Stores a unit test function with its name and type.
 *
 * NOTE: Every setup function must be paired with a teardown function. It's
 * possible to specify NULL function pointers.
 */
typedef struct UnitTest {
    const char* name;
    UnitTestFunction function;
    UnitTestFunctionType function_type;
} UnitTest;

typedef struct GroupTest {
    UnitTestFunction setup;
    UnitTestFunction teardown;
    const UnitTest *tests;
    const size_t number_of_tests;
} GroupTest;

/* Function prototype for test functions. */
```

```
typedef void (*CMUnitTestFunction)(void **state);

/* Function prototype for setup and teardown functions. */
typedef int (*CMFixtureFunction)(void **state);

struct CMUnitTest {
    const char *name;
    CMUnitTestFunction test_func;
    CMFixtureFunction setup_func;
    CMFixtureFunction teardown_func;
    void *initial_state;
};

/* Location within some source code. */
typedef struct SourceLocation {
    const char* file;
    int line;
} SourceLocation;

/* Event that's called to check a parameter value. */
typedef struct CheckParameterEvent {
    SourceLocation location;
    const char *parameter_name;
    CheckParameterValue check_value;
    LargestIntegralType check_value_data;
} CheckParameterEvent;

/* Used by expect_assert_failure() and mock_assert(). */
extern int global_expecting_assert;
```

```
extern jmp_buf global_expect_assert_env;
extern const char * global_last_failed_assert;

/* Retrieves a value for the given function, as set by "will_return". */
LargestIntegralType _mock(const char * const function, const char* const file,
                           const int line);

void _expect_function_call(
    const char * const function_name,
    const char * const file,
    const int line,
    const int count);

void _function_called(const char * const function, const char* const file,
                      const int line);

void _expect_check(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const CheckParameterValue check_function,
    const LargestIntegralType check_data, CheckParameterEvent * const event,
    const int count);

void _expect_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType values[],
    const size_t number_of_values, const int count);

void _expect_not_in_set(
    const char* const function, const char* const parameter,
```

```
const char* const file, const int line, const LargestIntegralType values[],  
const size_t number_of_values, const int count);  
  
void _expect_in_range(  
    const char* const function, const char* const parameter,  
    const char* const file, const int line,  
    const LargestIntegralType minimum,  
    const LargestIntegralType maximum, const int count);  
  
void _expect_not_in_range(  
    const char* const function, const char* const parameter,  
    const char* const file, const int line,  
    const LargestIntegralType minimum,  
    const LargestIntegralType maximum, const int count);  
  
void _expect_value(  
    const char* const function, const char* const parameter,  
    const char* const file, const int line, const LargestIntegralType value,  
    const int count);  
  
void _expect_not_value(  
    const char* const function, const char* const parameter,  
    const char* const file, const int line, const LargestIntegralType value,  
    const int count);  
  
void _expect_string(  
    const char* const function, const char* const parameter,  
    const char* const file, const int line, const char* string,  
    const int count);  
  
void _expect_not_string(  
    const char* const function, const char* const parameter,
```



```
    const char * const expression,
    const char * const file,
    const int line);

void _assert_int_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);

void _assert_int_not_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);

void _assert_string_equal(const char * const a, const char * const b,
    const char * const file, const int line);

void _assert_string_not_equal(const char * const a, const char * const b,
    const char *file, const int line);

void _assert_memory_equal(const void * const a, const void * const b,
    const size_t size, const char* const file,
    const int line);

void _assert_memory_not_equal(const void * const a, const void * const b,
    const size_t size, const char* const file,
    const int line);

void _assert_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);

void _assert_not_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);

void _assert_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line);

void _assert_not_in_set()
```



```

/* Standard output and error print methods. */

void print_message(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void print_error(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void vprint_message(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);
void vprint_error(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);

enum cm_message_output {
    CM_OUTPUT_STDOUT,
    CM_OUTPUT_SUBUNIT,
    CM_OUTPUT_TAP,
    CM_OUTPUT_XML,
};

/** @brief Function to set the output format for a test.

 * The ouput format for the test can either be set globally using this
 * function or overriden with environment variable CMOCKA_MESSAGE_OUTPUT.
 *
 * The environment variable can be set to either STDOUT, SUBUNIT, TAP or XML.
 *
 * @param[in] output  The output format to use for the test.
 *
 */
void cmocka_set_message_output(enum cm_message_output output);

/** @} */

#endif /* CMOCKA_H_ */

```

```
#include <sys/types.h>
#include <stdint.h>
#include <stddef.h>

#define KEY 0x123456

#undef KEY

#if defined(__i386)
# define KEY '_','_','i','3','8','6'
#elif defined(__x86_64)
# define KEY '_','_','x','8','6','_','6','4'
#elif defined(__ppc__)
# define KEY '_','_','p','p','c','_','_'
#elif defined(__ppc64__)
# define KEY '_','_','p','p','c','6','4','_','_'
#endif

#define SIZE (sizeof(unsigned short))

char info_size[] = {'I', 'N', 'F', 'O', ':', 's', 'i', 'z', 'e', '[',
('0' + ((SIZE / 10000)%10)),
('0' + ((SIZE / 1000)%10)),
('0' + ((SIZE / 100)%10)),
('0' + ((SIZE / 10)%10)),
('0' + (SIZE % 10)),
']',
#endif KEY
',','k','e','y','[',KEY,']',
#endif
'\0');
```

```
#ifdef __CLASSIC_C__  
  
int main(argc, argv) int argc; char *argv[];  
  
#else  
  
int main(int argc, char *argv[])  
  
#endif  
  
{  
  
    int require = 0;  
  
    require += info_size[argc];  
  
    (void)argv;  
  
    return require;  
  
}  
  
  
const char features[] = {"\n"  
"C_FEATURE:"  
  
#if (__GNUC__ * 100 + __GNUC_MINOR__) >= 404  
"1"  
  
#else  
"0"  
  
#endif  
  
"c_function_prototypes\n"  
"C_FEATURE:"  
  
#if (__GNUC__ * 100 + __GNUC_MINOR__) >= 404 && defined(__STDC_VERSION__) &&  
__STDC_VERSION__ >= 199901L  
"1"  
  
#else  
"0"  
  
#endif  
  
"c_restrict\n"  
"C_FEATURE:"
```

```

#if (__GNUC__ * 100 + __GNUC_MINOR__) >= 406 && defined(__STDC_VERSION__) &&
__STDC_VERSION__ >= 201000L
"1"

#else
"0"

#endif

"c_static_assert\n"
"C_FEATURE:"

#if (__GNUC__ * 100 + __GNUC_MINOR__) >= 404 && defined(__STDC_VERSION__) &&
__STDC_VERSION__ >= 199901L
"1"

#else
"0"

#endif

"c_variadic_macros\n"

};


```

```

int main(int argc, char** argv) { (void)argv; return features[argc]; }

#ifndef __cplusplus
# error "A C++ compiler has been selected for C."
#endif


```

```

#if defined(__18CXX)
#define ID_VOID_MAIN

#endif

#if defined(__CLASSIC_C__)
/* cv-qualifiers did not exist in K&R C */
#define const

```

```

#define volatile
#endif

/* Version number components: V=Version, R=Revision, P=Patch
   Version date components: YYYY=Year, MM=Month, DD=Day */

#if defined(__INTEL_COMPILER) || defined(__ICC)
#define COMPILER_ID "Intel"
#if defined(_MSC_VER)
#define SIMULATE_ID "MSVC"
#endif
/* __INTEL_COMPILER = VRP */
#define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
#define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
#if defined(__INTEL_COMPILER_UPDATE)
#define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
#else
#define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER % 10)
#endif
#if defined(__INTEL_COMPILER_BUILD_DATE)
/* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
#endif
#if defined(_MSC_VER)
/* _MSC_VER = VVRR */
#define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif

```

```

#elif defined(__PATHCC__)

#define COMPILER_ID "PathScale"

#define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
#define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
#if defined(__PATHCC_PATCHLEVEL__)
#define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
#endif

#endif /* !defined(COMPILER_ID) */

#ifndef COMPILER_ID
#define COMPILER_ID "Unknown"
#endif

/* Compiler detection code */

#if defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)

#define COMPILER_ID "Embarcadero"
#define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__>>24 & 0x00FF)
#define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__>>16 & 0x00FF)
#define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__ & 0xFFFF)

#endif /* __BORLANDC__ && __CODEGEARC_VERSION__ */

#if defined(__BORLANDC__)

#define COMPILER_ID "Borland"
/* __BORLANDC__ = 0xVRR */
#define COMPILER_VERSION_MAJOR HEX(__BORLANDC__>>8)
#define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#endif /* __BORLANDC__ */

#if defined(__WATCOMC__) && __WATCOMC__ < 1200

#define COMPILER_ID "Watcom"
/* __WATCOMC__ = VVRR */
#define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
#define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
#if (__WATCOMC__ % 10) > 0
#define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
#endif

#endif /* __WATCOMC__ < 1200 */
#endif /* !defined(COMPILER_ID) */

```

```
#elif defined(__WATCOMC__)

# define COMPILER_ID "OpenWatcom"

/* __WATCOMC__ = VVRP + 1100 */

# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ - 1100) / 100

# define COMPILER_VERSION_MINOR DEC(__WATCOMC__ / 10) % 10

# if (__WATCOMC__ % 10) > 0

# define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)

# endif


#elif defined(__SUNPRO_C)

# define COMPILER_ID "SunPro"

# if __SUNPRO_C >= 0x5100

/* __SUNPRO_C = 0xVRRP */

# define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C>>12)

# define COMPILER_VERSION_MINOR HEX(__SUNPRO_C>>4 & 0xFF)

# define COMPILER_VERSION_PATCH HEX(__SUNPRO_C & 0xF)

# else

/* __SUNPRO_CC = 0xVRP */

# define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C>>8)

# define COMPILER_VERSION_MINOR HEX(__SUNPRO_C>>4 & 0xF)

# define COMPILER_VERSION_PATCH HEX(__SUNPRO_C & 0xF)

# endif


#elif defined(__HP_cc)

# define COMPILER_ID "HP"

/* __HP_cc = VVRRPP */

# define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)

# define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
```

```

#define COMPILER_VERSION_PATCH DEC(__HP_cc % 100)

#if defined(__DECC)
#define COMPILER_ID "Compaq"
/* __DECC_VER = VVRRTPPPP */

#define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
#define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000 % 100)
#define COMPILER_VERSION_PATCH DEC(__DECC_VER % 10000)

#elif defined(__IBMC) && defined(__COMPILER_VER__)
#define COMPILER_ID "zOS"
/* __IBMC__ = VRP */

#define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
#define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
#define COMPILER_VERSION_PATCH DEC(__IBMC__ % 10)

#elif defined(__IBMC) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
#define COMPILER_ID "XL"
/* __IBMC__ = VRP */

#define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
#define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
#define COMPILER_VERSION_PATCH DEC(__IBMC__ % 10)

#elif defined(__IBMC) && !defined(__COMPILER_VER__) && __IBMC__ < 800
#define COMPILER_ID "VisualAge"
/* __IBMC__ = VRP */

#define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
#define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
#define COMPILER_VERSION_PATCH DEC(__IBMC__ % 10)

```

```
#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
# define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
/* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000 % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__ % 1000)

#elif defined(__FUJITSU) || defined(__FCC_VERSION) || defined(__fcc_version)
# define COMPILER_ID "Fujitsu"

#elif defined(__TINYC__)
# define COMPILER_ID "TinyCC"

#elif defined(__BCC__)
# define COMPILER_ID "Bruce"
```

```
#elif defined(__SCO_VERSION__)

#define COMPILER_ID "SCO"

#elif defined(__clang__) && defined(__apple_build_version__)

#define COMPILER_ID "AppleClang"

#if defined(_MSC_VER)

#define SIMULATE_ID "MSVC"

#endif

#define COMPILER_VERSION_MAJOR DEC(__clang_major__)

#define COMPILER_VERSION_MINOR DEC(__clang_minor__)

#define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)

#if defined(_MSC_VER)

/* _MSC_VER = VVRR */

#define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)

#define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)

#endif

#define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__)

#define COMPILER_ID "Clang"

#if defined(_MSC_VER)

#define SIMULATE_ID "MSVC"

#endif

#define COMPILER_VERSION_MAJOR DEC(__clang_major__)

#define COMPILER_VERSION_MINOR DEC(__clang_minor__)

#define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)

#if defined(_MSC_VER)

/* _MSC_VER = VVRR */


```

```

# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__GNUC__)
# define COMPILER_ID "GNU"
# define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# if defined(__GNUC_MINOR__)
# define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
# define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
/* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
# if _MSC_VER >= 1400
/* _MSC_FULL_VER = VVRRPPPP */
# define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
# else
/* _MSC_FULL_VER = VVRRPPPP */
# define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
# endif
# endif
# if defined(_MSC_BUILD)

```



```

#endif /* _COMPILER_ID */

#ifndef _COMPILER_ID
#define _COMPILER_ID "Unknown Compiler"
#endif /* _COMPILER_ID */

/* These compilers are either not known or too old to define an
identification macro. Try to identify the platform and guess that
it is the native compiler. */

#if defined(SDCC)
#define COMPILER_ID "SDCC"
/* SDCC = VRP */
#define COMPILER_VERSION_MAJOR DEC(SDCC/100)
#define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
#define COMPILER_VERSION_PATCH DEC(SDCC % 10)

#elif defined(_SGI_COMPILER_VERSION) || defined(_COMPILER_VERSION)
#define COMPILER_ID "MIPSpro"
#if defined(_SGI_COMPILER_VERSION)
/* _SGI_COMPILER_VERSION = VRP */
#define COMPILER_VERSION_MAJOR DEC(_SGI_COMPILER_VERSION/100)
#define COMPILER_VERSION_MINOR DEC(_SGI_COMPILER_VERSION/10 % 10)
#define COMPILER_VERSION_PATCH DEC(_SGI_COMPILER_VERSION % 10)
#else
/* _COMPILER_VERSION = VRP */
#define COMPILER_VERSION_MAJOR DEC(_COMPILER_VERSION/100)
#define COMPILER_VERSION_MINOR DEC(_COMPILER_VERSION/10 % 10)
#define COMPILER_VERSION_PATCH DEC(_COMPILER_VERSION % 10)
#endif
#endif /* _COMPILER_ID */

#endif /* _COMPILER_ID */

```

```

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)

/* Identify known platforms by name. */

#ifndef _LIBINTL_H
#define _LIBINTL_H

/* Construct the string literal in pieces to prevent the source from
   getting matched. Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array. */

char const* info_compiler = "INFO ":"compiler[" COMPILER_ID "]";
#endif /* _LIBINTL_H */

#ifndef _LIBINTL_H
#define _LIBINTL_H

/* Construct the string literal in pieces to prevent the source from
   getting matched. Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array. */

char const* info_simulate = "INFO ":"simulate[" SIMULATE_ID "]";
#endif /* _LIBINTL_H */

#ifndef _LIBINTL_H
#define _LIBINTL_H

/* Construct the string literal in pieces to prevent the source from
   getting matched. Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array. */

char const* qnxnto = "INFO ":"qnxnto[]";
#endif /* _LIBINTL_H */

#ifndef _LIBINTL_H
#define _LIBINTL_H

/* Construct the string literal in pieces to prevent the source from
   getting matched. Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array. */

char const *info_cray = "INFO ":"compiler_wrapper[CrayPrgEnv]";
#endif /* _LIBINTL_H */

```

```
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__)
|| defined(_aix) || defined(__aix__)
# define PLATFORM_ID "AIX"
```

```
#elif defined(__sgi) || defined(__sgi__) || defined(_SGI)
# define PLATFORM_ID "IRIX"

#elif defined(__hpx) || defined(__hpx__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
```

```
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
#if defined(__LINUX__)
# define PLATFORM_ID "Linux"

#elif defined(__DOS__)
# define PLATFORM_ID "DOS"

#elif defined(__OS2__)
# define PLATFORM_ID "OS2"

#elif defined(__WINDOWS__)
# define PLATFORM_ID "Windows3x"
```

```
# else /* unknown platform */

# define PLATFORM_ID

# endif

#else /* unknown platform */

# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
the architecture of the compiler being used. This is because
the compilers do not have flags that can change the architecture,
but rather depend on which compiler is being used

*/
#ifndef _MSC_VER
# if defined(_M_IA64)
# define ARCHITECTURE_ID "IA64"

# elif defined(_M_X64) || defined(_M_AMD64)
# define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
# define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM)
# if _M_ARM == 4
# define ARCHITECTURE_ID "ARMV4I"
# elif _M_ARM == 5
```

```
# define ARCHITECTURE_ID "ARMV5I"

# else

# define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)

# endif


# elif defined(_M_MIPS)

# define ARCHITECTURE_ID "MIPS"


# elif defined(_M_SH)

# define ARCHITECTURE_ID "SHx"


# else /* unknown architecture */

# define ARCHITECTURE_ID ""

# endif


#elif defined(__WATCOMC__)

# if defined(_M_I86)

# define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)

# define ARCHITECTURE_ID "X86"

# else /* unknown architecture */

# define ARCHITECTURE_ID ""

# endif


#else

# define ARCHITECTURE_ID

#endif
```

```

/* Convert integer to decimal digit literals. */

#define DEC(n)      \
('0' + (((n) / 10000000)%10)), \
('0' + (((n) / 1000000)%10)), \
('0' + (((n) / 100000)%10)), \
('0' + (((n) / 10000)%10)), \
('0' + (((n) / 1000)%10)), \
('0' + (((n) / 100)%10)), \
('0' + (((n) / 10)%10)), \
('0' + ((n) % 10))

/* Convert integer to hex digit literals. */

#define HEX(n)      \
('0' + ((n)>>28 & 0xF)), \
('0' + ((n)>>24 & 0xF)), \
('0' + ((n)>>20 & 0xF)), \
('0' + ((n)>>16 & 0xF)), \
('0' + ((n)>>12 & 0xF)), \
('0' + ((n)>>8  & 0xF)), \
('0' + ((n)>>4  & 0xF)), \
('0' + ((n)    & 0xF))

/* Construct a string literal encoding the version number components. */

#ifndef COMPILER_VERSION_MAJOR
char const info_version[] = {
'I', 'N', 'F', 'O', ':',
'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
COMPILER_VERSION_MAJOR,

```

```

#define COMPILER_VERSION_MINOR
    '.', COMPILER_VERSION_MINOR,
#define COMPILER_VERSION_PATCH
    '.', COMPILER_VERSION_PATCH,
#define COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#endif
#endif
#endif
'']','\0'};

/* Construct a string literal encoding the version number components. */
#ifndef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
    'I', 'N', 'F', 'O', ':',
    's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
SIMULATE_VERSION_MAJOR,
#ifndef SIMULATE_VERSION_MINOR
    '.', SIMULATE_VERSION_MINOR,
#endif
#ifndef SIMULATE_VERSION_PATCH
    '.', SIMULATE_VERSION_PATCH,
#endif
#ifndef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#endif
'']','\0'};

#endif
#endif
#endif
'']','\0'};

#endif

```

```
/* Construct the string literal in pieces to prevent the source from
   getting matched. Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array. */

char const* info_platform = "INFO ":"platform[" PLATFORM_ID "]";
char const* info_arch = "INFO ":"arch[" ARCHITECTURE_ID "]";

#if !defined(__STDC__)
# if defined(_MSC_VER) && !defined(__clang__)
# define C_DIALECT "90"
# else
# define C_DIALECT
# endif
#elif __STDC_VERSION__ >= 201000L
# define C_DIALECT "11"
#elif __STDC_VERSION__ >= 199901L
# define C_DIALECT "99"
#else
# define C_DIALECT "90"
#endif

const char* info_language_dialect_default =
    "INFO ":"dialect_default[" C_DIALECT "]";

/*-----*/
```

```
#ifdef ID_VOID_MAIN
void main() {}

#else
# if defined(__CLASSIC_C__)
int main(argc, argv) int argc; char *argv[];
# else
int main(int argc, char* argv[])
# endif
{
    int require = 0;
    require += info_compiler[argc];
    require += info_platform[argc];
    require += info_arch[argc];
#endifdef COMPILER_VERSION_MAJOR
    require += info_version[argc];
#endif
#endifdef SIMULATE_ID
    require += info_simulate[argc];
#endif
#endifdef SIMULATE_VERSION_MAJOR
    require += info_simulate_version[argc];
#endif
#endifdef __CRAYXE || defined(__CRAYXC)
    require += info_cray[argc];
#endif
    require += info_language_dialect_default[argc];
    (void)argv;
    return require;
}
```

```
#endif

/* Name of package */

#define PACKAGE "cmocka"

/* Version number of package */

#define VERSION "1.1.0"

/* #undef LOCALEDIR */

#define DATADIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/build-
Debug/share/cmocka"

#define LIBDIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/build-Debug/lib"

#define PLUGINDIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/build-
Debug/lib/cmocka-0"

#define SYSCONFDIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/build-
Debug/etc"

#define BINARYDIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/build-
Debug/cmocka"

#define SOURCEDIR "/home/harsimransingh/ecn5013/tutorials/unit_tests/3rd-party/cmocka"

***** HEADER FILES *****

/* Define to 1 if you have the <assert.h> header file. */

#define HAVE_ASSERT_H 1

/* Define to 1 if you have the <dlfcn.h> header file. */

/* #undef HAVE_DLFCN_H */

/* Define to 1 if you have the <inttypes.h> header file. */

#define HAVE_INTTYPES_H 1
```

```
/* Define to 1 if you have the <io.h> header file. */
/* #undef HAVE_IO_H */

/* Define to 1 if you have the <malloc.h> header file. */
#define HAVE_MALLOC_H 1

/* Define to 1 if you have the <memory.h> header file. */
#define HAVE_MEMORY_H 1

/* Define to 1 if you have the <setjmp.h> header file. */
#define HAVE_SETJMP_H 1

/* Define to 1 if you have the <signal.h> header file. */
#define HAVE_SIGNAL_H 1

/* Define to 1 if you have the <stdarg.h> header file. */
#define HAVE_STDARG_H 1

/* Define to 1 if you have the <stddef.h> header file. */
#define HAVE_STDDEF_H 1

/* Define to 1 if you have the <stdint.h> header file. */
#define HAVE_STDINT_H 1

/* Define to 1 if you have the <stdio.h> header file. */
#define HAVE_STDIO_H 1

/* Define to 1 if you have the <stdlib.h> header file. */
#define HAVE_STDLIB_H 1
```

```
/* Define to 1 if you have the <strings.h> header file. */
#define HAVE_STRINGS_H 1

/* Define to 1 if you have the <string.h> header file. */
#define HAVE_STRING_H 1

/* Define to 1 if you have the <sys/stat.h> header file. */
#define HAVE_SYS_STAT_H 1

/* Define to 1 if you have the <sys/types.h> header file. */
#define HAVE_SYS_TYPES_H 1

/* Define to 1 if you have the <time.h> header file. */
#define HAVE_TIME_H 1

/* Define to 1 if you have the <unistd.h> header file. */
#define HAVE_UNISTD_H 1

***** STRUCTS *****

#define HAVE_STRUCT_TIMESPEC 1

***** FUNCTIONS *****

/* Define to 1 if you have the `calloc' function. */
/* #undef HAVE_CALLOC */

/* Define to 1 if you have the `exit' function. */
```

```
/* #undef HAVE_EXIT */

/* Define to 1 if you have the `fprintf' function. */
/* #undef HAVE_FPRINTF */

/* Define to 1 if you have the `snprintf' function. */
/* #undef HAVE_SNPRINTF */

/* Define to 1 if you have the `_snprintf' function. */
/* #undef HAVE__SNPRINTF */

/* Define to 1 if you have the `_snprintf_s' function. */
/* #undef HAVE__SNPRINTF_S */

/* Define to 1 if you have the `vsnprintf' function. */
/* #undef HAVE_VSNPRINTF */

/* Define to 1 if you have the `_vsnprintf' function. */
/* #undef HAVE__VSNPRINTF */

/* Define to 1 if you have the `_vsnprintf_s' function. */
/* #undef HAVE__VSNPRINTF_S */

/* Define to 1 if you have the `free' function. */
/* #undef HAVE_FREE */

/* Define to 1 if you have the `longjmp' function. */
#define HAVE_LONGJMP 1
```

```
/* Define to 1 if you have the `siglongjmp' function. */
#define HAVE_SIGLONGJMP 1
```

```
/* Define to 1 if you have the `malloc' function. */
/* #undef HAVE_MALLOC */
```

```
/* Define to 1 if you have the `memcpy' function. */
/* #undef HAVE_MEMCPY */
```

```
/* Define to 1 if you have the `memset' function. */
/* #undef HAVE_MEMSET */
```

```
/* Define to 1 if you have the `printf' function. */
/* #undef HAVE_PRINTF */
```

```
/* Define to 1 if you have the `setjmp' function. */
#define HAVE_SETJMP 1
```

```
/* Define to 1 if you have the `signal' function. */
#define HAVE_SIGNAL 1
```

```
/* Define to 1 if you have the `snprintf' function. */
/* #undef HAVE_SNPRINTF */
```

```
/* Define to 1 if you have the `strcmp' function. */
/* #undef HAVE_STRCMP */
```

```
/* Define to 1 if you have the `strcpy' function. */
/* #undef HAVE_STRCPY */
```

```
/* Define to 1 if you have the `vsnprintf' function. */
/* #undef HAVE_VSNPRINTF */

/* Define to 1 if you have the `strsignal' function. */
#define HAVE_STRSIGNAL 1

/* Define to 1 if you have the `clock_gettime' function. */
#define HAVE_CLOCK_GETTIME 1

/********************* OPTIONS ********************/

/* Check if we have TLS support with GCC */
#define HAVE_GCC_THREAD_LOCAL_STORAGE 1

/* Check if we have TLS support with MSVC */
/* #undef HAVE_MSVC_THREAD_LOCAL_STORAGE */

/* Check if we have CLOCK_REALTIME for clock_gettime() */
#define HAVE_CLOCK_GETTIME_REALTIME 1

/********************* ENDIAN *******************/

#define WORDS_SIZEOF_VOID_P 8

/* Define WORDS_BIGENDIAN to 1 if your processor stores words with the most
   significant byte first (like Motorola and SPARC, unlike Intel). */
/* #undef WORDS_BIGENDIAN */

/*********************
```

```
* * Authors : Harsimransingh Bindra, Smitesh Modak
* * Date : 10/27/2017
*
* * File : test_memory.c
* * Description : Source file for testing memory functions
* * -invalid_pointer_test_mymemmove()
* * -no_overlap_test_mymemmove()
* * -SRC_in_DST_test_mymemmove()
* * -DST_in_SRC_test_memmove()
* * -invalid_pointer_test_mymemset()
* * -valid_test_mymemset()
* * -invalid_pointer_test_mymemset()
* * -valid_test_mymemset()
* * -invalid_pointer_test_reverse()
* * -odd_length_test_myreverse()
* * -even_length_test_myreverse()
* * -character256_test_myreverse()
*
* * Testing environment : To test the functions we need
* * to install the Cmocka framework libraries in our folder
* * path and make changes to the test Makefile to include
* * our source files and test files.

* *****/
#include <stdio.h>
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <stdint.h>
#include "memory.h"
```

```

#include <cmocka.h>

/***********************
 * * Defining global variables required for the test function
 * * even_length and odd_length to provide odd and even lengths
 * * test_set[32] array to behave as a memory
 * *****/
int32_t even_length = 10;
uint32_t odd_length = 11;
uint8_t test_set[32] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32};

/***********************
 * * invalid_pointer_test_mymemmove() - This function is used to test NULL pointer
 * * check in mymemmove function.
 * * Input parameter - For this we can send either source or dest as
 * * NULL and some length
 * * Expected Result - Since an invalid(NULL) pointer is passed as a
 * * parameter the function should return an Invalid_pointer
 * * enum
 * *****/
void invalid_pointer_test_mymemmove(void **state)
{
    uint8_t *source = &test_set[0];
    uint8_t *dest = NULL;
    memory_status_t status = my_memmove(source,dest,even_length);
    assert_int_equal(status, Invalid_pointer);
}

```

```

/***********************/

* * no_overlap_test_mymemmove() - This function is used to test my_memmove function
* * to move data from source to dest memory locations without overlap
* * Input parameter - For this we send a pointer to valid src and dst
* * locations along with the length in bytes so they don't overlap
* * Expected Result - Since it does not have an invalid length or
* * invalid pointer it should successfully move memory and return Pass
* * enum value.

* *****/
void no_overlap_test_mymemmove(void **state)
{
    uint8_t *source = &test_set[5];
    uint8_t *dest = &test_set[20];
    memory_status_t status = my_memmove(source,dest,even_length);
    assert_int_equal(status, Pass);
}

/***********************/

* * SRC_in_DST_test_mymemmove() - This function is used to test mymemmove function
* * to move data from source to dest memory locations with source in dest
* * overlap
* * Input parameter - For this we send a pointer to valid source and dest
* * locations along with the length in bytes so that source is in dest
* * Assumption - src in dst memory case can be handled by the function
* * Expected Result - Since it does not have an invalid length or
* * invalid pointer it should successfully move memory with source in
* * dest and return Pass enum value.

* *****/

```

```

void SRC_in_DST_test_mymemmove(void **state)
{
    uint8_t *source = &test_set[15];
    uint8_t *dest = &test_set[10];
    memory_status_t status = my_memmove(source, dest, even_length);
    assert_int_equal(status, Pass);
}

/****************************************
 * * DST_in_SRC_test_mymemmove() - This function is used to test mymemmove function
 * * to move data from src to dst memory locations with dst in src
 * * overlap
 * * Input parameter - For this we send a pointer to valid source and dest
 * * locations along with the length in bytes so that dst is in src
 * * Expected Result - Since it does not have an invalid length or
 * * invalid pointer it should successfully move memory with dest in
 * * source and return Pass enum value.
****************************************/
void DST_in_SRC_test_mymemmove(void **state)
{
    uint8_t *source = &test_set[10];
    uint8_t *dest = &test_set[15];
    memory_status_t status = my_memmove(source, dest, even_length);
    assert_int_equal(status, Pass);
}

/****************************************
 * * invalid_pointer_test_mymemset() - Function to test NULL pointer
 * * check in mymemset function.
****************************************/

```

* * Input parameter - For this we send source as NULL with some value

* * and length

* * Expected Result - Since an invalid(NULL) pointer is passed as a

* * parameter the function should return an Invalid_pointer

* * enum

* ******/

```
void invalid_pointer_test_mymemset(void **state)
{
    uint8_t *source = NULL;
    uint8_t set_value = 10;
    memory_status_t status = my_memset(source, even_length, set_value);
    assert_int_equal(status, Invalid_pointer);
}
```

* * valid_test_mymemset() - This function is used to test mymemset function to

* * set a particular value in memory

* * Input parameter - For this we send a pointer to a valid memory

* * location along with the length in bytes and value to set

* * Expected Result - Since it does not have an invalid length or

* * invalid pointer it should successfully set value and return

* * Pass enum value.

* ******/

```
void valid_test_mymemset(void **state)
```

```
{
```

```
    uint8_t *source = &test_set[0];
```

```
    uint8_t test_value = 10;
```

```

memory_status_t status = my_memset(source,even_length,test_value);

assert_int_equal(status, Pass);

}

/**********************/

* * invalid_pointer_test_mymemzero() - This function is used to test NULL pointer
* * check in mymemzero function.

* * Input parameter - For this we send src as NULL with some length
* * Expected Result - Since an invalid(NULL) pointer is passed as a
* * parameter the function should return an Invalid_pointer
* * enum vlaue
* ****

void invalid_pointer_test_mymemzero(void **state)
{
    uint8_t *source = NULL;

    memory_status_t status = my_memzero(source,even_length);
    assert_int_equal(status, Invalid_pointer);
}

/**********************/

* * valid_test_mymemzero() - This function to test mymemzero function to
* * set zeros in memory
* * Input parameter - For this we send a pointer to a valid memory
* * location along with the length in bytes
* * Expected Result - Since it does not have an invalid length or
* * invalid pointer it should successfully set zeros and return Pass
* * enum value.
* ****

void valid_test_mymemzero(void **state)

```

```

{
    uint8_t *source = &test_set[0];
    memory_status_t status = my_memzero(source,even_length);
    assert_int_equal(status, Pass);
}

/***********************
 * * invalid_pointer_test_myreverse() - This function is used to test NULL pointer
 * * check in myreverse function.
 * * Input parameter - For this we can send source as NULL and some
 * * length
 * * Expected Result - Since an invalid(NULL) pointer is passed as a
 * * parameter the function should return an Invalid_pointer
 * * emum value.
 * *****/
void invalid_pointer_test_myreverse(void **state)
{
    uint8_t *source = NULL;
    memory_status_t status = my_reverse(source,even_length);
    assert_int_equal(status, Invalid_pointer);
}

/***********************
 * * odd_length_test_myreverse() - This function is used to test myreverse function with
 * * odd length
 * * Input parameter - For this we send a pointer to a valid memory
 * * location along with an odd length in bytes
 * * Assumption: Function is written to handle odd length cases as
 * * well

```

```

* * Expected Result - Since it does not have an invalid length or
* * invalid pointer it should successfully reverse odd memory and
* * return Pass enum value.

* ****
void odd_length_test_myreverse(void **state)
{
    uint8_t *source = &test_set[10];
    memory_status_t status = my_reverse(source,odd_length);
    assert_int_equal(status, Pass);
}

****

* * even_length_test_myreverse() - This function is used to test myreverse function with
* * even length
* * Input parameter - For this we send a pointer to a valid memory
* * location along with an even length in bytes
* * Expected Result - Since it does not have an invalid length or
* * invalid pointer it should successfully reverse even memory and
* * return Pass enum value.

* ****
void even_length_test_myreverse(void **state)
{
    uint8_t *source = &test_set[10];
    memory_status_t status = my_reverse(source,even_length);
    assert_int_equal(status, Pass);
}

****

* * character256_test_myreverse() - This function is used to test myreverse function

```

```

* * with all characters

* * Input parameter - For this we send a pointer to a memory with

* * all 256 characters

* * Expected Result - Since it does not have an invalid length or

* * invalid pointer it should successfully reverse memory and

* * return Pass enum value.

* *****/
void character256_test_myreverse(void **state)
{
    uint8_t test_set[256];
    uint8_t test_check[256];
    uint16_t i;
    for(i=0;i<256;i++)
    {
        test_set[i] = i;
        test_check[i] = i;
    }
    uint8_t *source = test_set;
    memory_status_t status = my_reverse(source,256);
    for(i=0;i<256;i++)
    {
        if(test_set[i]!=test_check[255-i])
        {
            status = Fail;
            break;
        }
    }
    if(i == 256)
        assert_int_equal(status, Pass);
}

```

```
else assert_int_equal(status, Fail);

}

int main(int argc, char **argv)
{
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(invalid_pointer_test_mymemmove),
        cmocka_unit_test(no_overlap_test_mymemmove),
        cmocka_unit_test(SRC_in_DST_test_mymemmove),
        cmocka_unit_test(DST_in_SRC_test_mymemmove),
        cmocka_unit_test(invalid_pointer_test_my memset),
        cmocka_unit_test(valid_test_my memset),
        cmocka_unit_test(invalid_pointer_test_my memset),
        cmocka_unit_test(valid_test_my memset),
        cmocka_unit_test(invalid_pointer_test_myreverse),
        cmocka_unit_test(odd_length_test_myreverse),
        cmocka_unit_test(even_length_test_myreverse),
        cmocka_unit_test(character256_test_myreverse),
    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include "circbuf.h"

void main()
{
    uint8_t retval,retval1;
    uint8_t i;
```

```
CB_status status = Available;
CB_t *cbuf = malloc(sizeof(CB_t));
CB_init(cbuf, 6);
for(i=0; i<6; i++)
{
    CB_buffer_add_item(cbuf, i);
}
printf("buffer size is:%d\n",cbuf->count);
CB_buffer_remove_item(cbuf);
CB_buffer_remove_item(cbuf);
printf("buffer size is:%d\n",cbuf->count);
//for(i=0; i<3; i++)
//{
//    CB_buffer_add_item(cbuf, i);
//}
//for(i=0; i<=4; i++)
//{
//    CB_buffer_remove_item(cbuf);
//}
status = CB_peek(cbuf, 3);
printf("peek value is %d\n", (cbuf->peek));
if(status == No_error)
    printf("hi\n");
printf("buffer size is: %d\n",cbuf->count);

for (i=0; i<6; i++)
{
    printf("%d\n", *(cbuf->buffer+i));
}
```

```

status=CB_destroy(cbuf);

if(status == No_error)

printf("hi\n");

//printf("%d\n", *(cbuf->buffer+1));

}

*****
* * Authors: Harsimransingh Bindra, Smitesh Modak
* * Date Edited: 10/01/2017
* *
* * File: memory.h
* *
* * Description: Header file for data manipulation
* * -cal_length
* * -my_memmove
* * -my_memcpy
* * -my_memset
* * -my_memzero
* * -my_reverse
* * -reserve_words
* * -free_words
* *
* *
* ****
header for function called "memory"*/

```

```

#ifndef MEMORY_H_
#define MEMORY_H_
#include <stdio.h>

```

```

#include <stdint.h>

typedef enum status_memory{
    Pass,
    Invalid_pointer,
    Fail
}memory_status_t;

uint8_t cal_length(uint8_t *ptr);

/**
 * Function to calculate length of the string
 *
 * This function calculates the length of the string which is stored at the memory location pointed by the
 * pointer passed
 *
 * returns uint8_t(length) */
}

uint8_t my_memmove(uint8_t *src, uint8_t *dst,size_t length);

/**
 * Function to move a data from one location to another
 *
 * This function is used to move data from one memory location to another.
 * overlapping of the two memory locations is checked and handled accordingly
 *
 * returns uint8_t */
}

uint8_t my_memcpy(uint8_t *src, uint8_t *dst,size_t length);

/**
 * Function to copy data from one location to another
 *

```

```
* This function is used to copy data from one memory location to another
* This function does not handle overlapping and data may be overwritten
*
* returns uint8_t*/
uint8_t my_memset(uint8_t *src, size_t length, uint8_t value);

/***
* Function to set particular value to all given locations
*
* This function sets a particular value given as parameter to all given memory locations.
*
* returns uint8_t*/
uint8_t my_memzero(uint8_t *src, size_t length);

/***
* function to set all values to 0 at given memory locations
*
* This function is a more specific version of memset function, it sets the value to 0 for all the memory
location given
*
* returns uint8_t*/
uint8_t my_reverse(uint8_t *src, size_t length);

/***
* Function to reverse data at given memory location
*
* This function reverses a particular array or a given string using pointer arithematic
*
* returns uint8_t*/
uint32_t *reserve_words(size_t length);

/***
* function to reserve a particular memory location through dynamic memory allocation
```

```

*
* This function dynamically allocates memory to the pointer passed as the parameter
*
* returns uint32_t*/
void free_words(uint32_t *src);

/***
* Function to free dynamically allocated memory
*
* This function frees the memory which was dynamically allocated for the function. It avoids the
memory exhaustion
*
* returns void */
#endif

*****  

* * Authors : Harsimransingh Bindra, Smitesh Modak
* * Date : 10/28/2017
* *
* * File : circbuff.h
* * Description : header files for Circular buffer functions
* * CB_status CB_init(CB_t *cbuf,uint8_t length)
* * CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val)
* * CB_status CB_buffer_remove_item(CB_t *cbuf)
* * CB_status CB_is_full(CB_t *cbuf)
* * CB_status CB_is_empty(CB_t *cbuf)
* * CB_status CB_peek(CB_t *cbuf, uint8_t position)
* * CB_status CB_destroy(CB_t *cbuf)
* *
*****/

```

```
#include <stdint.h>

/*Enum used for returning the buffer state*/
typedef enum CB_Enum_t{
    No_error=1,
    Full,
    Empty,
    Available,
    Null_pointer,
    No_length,
    Buffer_allocation_failure,
    Wrong_memory_access,
    Error
}CB_status;

/*Circular buffer structure*/
typedef struct circular
{
    volatile uint8_t *buffer;
    volatile uint8_t *tail;
    volatile uint8_t *head;
    volatile uint8_t val;
    volatile uint8_t count;
    volatile uint8_t length;
    volatile uint8_t peek;
    CB_status status;
}CB_t;

/*********************
```

```

* * This function is used to add elements to circular buffer
* * The return value can be full or available
* * CB_t *cbuf - Pointer which points to circular buffer
* * uint8_t val - data to be stored on buffer
***** */

CB_status CB_buffer_add_item(CB_t *cbuf, uint8_t val);

void memcopy(uint8_t *tail, uint8_t val);

***** */

* * This function is used to allocate heap memory to circular buffer
* * CB_t *cbuf - pointer which points to the circular buffer
* * uint8_t length - This variable gives the length of circular buffer
***** /

CB_status CB_init(CB_t *cbuf,uint8_t length);

***** */

* * This function is used to remove elements from circular buffer
* * The return value can be empty or available depending on count value
* * CB_t *cbuf - Pointer which points to circular buffer
* **** */
CB_status CB_buffer_remove_item(CB_t *cbuf);

***** */

* * This function is used to check if circular buffer is full
* * The return value can be full or available depending on count value
* * CB_t *cbuf - Pointer which points to circular buffer
* **** */
CB_status CB_is_full(CB_t *cbuf);

```

```

/*****  

* * This function is used to check if circular buffer is empty  

* * The return value can be empty or available depending on count value  

* * CB_t *cbuf - Pointer which points to circular buffer  

* *****/  

CB_status CB_is_empty(CB_t *cbuf);  
  

/*****  

* * This function is to check the value at a particular location of buffer  

* * It returns the value present at that location  

* * CB_t* cbuf - Pointer which points to circular buffer  

* * uint8_t position - position to look for  

* *****/  

CB_status CB_peek(CB_t *cbuf, uint8_t position);  
  

/*****  

* * This function is used to free the heap memory allocated to circular buffer  

* * CB_t *cbuf - Pointer which points to circular buffer  

* *****/  

CB_status CB_destroy(CB_t *cbuf);  
  

/*****  

* * Authors: Harsimransingh Bindra, Smitesh Modak  

* * Date Edited: 10/01/2017  

* *  

* * File: debug.h  

* *  

* * Description: Header file for data manipulation

```

```
 * * -print_memory
 *
 *
 ****
*/header for function called "memory"/

#ifndef DEBUG_H_
#define DEBUG_H
#include<stdint.h>
void print_memory(uint8_t *start, uint32_t length);
/**
 * Function to print data present at the given memory location
 *
 * This function prints the data from the memory location to where the start pointer is pointing to
 * This function is used for debugging
 *
 * returns void*/
#endif
*****
 * * Authors: Harsimransingh Bindra, Smitesh Modak
 * * Date Edited: 10/01/2017
 *
 * * File: conversion.h
 *
 * * Description: Header file for data manipulation
 * * -my_itoa
 * * -my_atoi
 * * -dump_memory
 * * -big_to_little32
```

```

* * -little_to_big32
* *
* *
* ****
header file for function called "conversion"*/

```

```

#ifndef CONERSION_H_
#define CONVERSION_H_

typedef enum conversion_status{
    Invalid_pointer,
    Pass,
    Fail
}conversion_status_t;

uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base);
/***
 * Function converting integral values to strings
 *
 * This function takes an integer value as input and converts it into a string and handles both the signed
 * and unsigned data.
 *
 * It handles number from base 2 to base 16
 *
 * returns uint8_t*/
int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base);
/**Function converting string ascii inputs to integers
 *
 * This function takes a string as input and converts it into a integer value and handles both the signed
 * and unsigned data.

```

```

* It handles number from base 2 to base 16
*
* returns uint8_t*/
int8_t big_to_little32(uint32_t *data, uint32_t length);

/**
* Changing endianness from big endian to little endian
*
* This function takes data which is saved in big endian format and changes the endianness to little
endian.

* This function is used when the data needs to be transferred to different platforms with different
endianness

*
* returns int8_t*/
int8_t little_to_big32(uint32_t *data, uint32_t length);

/**
* Changing endianness from little endian to big endian
*
* This function takes data which is saved in little endian format and changes the endianness to big
endian.

* This function is used when the data needs to be transferred to different platforms with different
endianness

*
* returns int8_t*/

#endif
*****
* * Authors : Smitesh Modak, Harsimrarsingh
* * Date : 02/28/2017
* *
* * File : unittest_data.c

```

```

* * Description : Source file for testing data functions

* *      -invalid_test_big_to_little()

* *      -invalid_test_little_to_big()

* *      -valid_conversion_test_big_to_little()

* *      -valid_conversion_test_little_to_big()

* * Testing environment : To test the functions we need

* * to install the Cmocka framework libraries in our folder

* * path and make changes to the test Makefile to include

* * our source files and test files.

* *****/
#include<stdio.h>

#include<stdint.h>

#include"conversion.h"

#include <stdarg.h>

#include <stddef.h>

#include <setjmp.h>

#include <cmocka.h>

/**/

* * invalid_test_big_to_little() - Function to test NULL pointer

* * check in bit_to_little32 function.

* * Input parameter - For this we send the NULL pointer as an input

* * parameter along with some length

* * Expected Result - Since an invalid(NULL) pointer is passed as a

* * parameter the function should return an Invalid_pointer enum value

* *****/
void invalid_test_big_to_little(void **state)
{
    uint32_t *source = NULL;
    uint32_t test_length = 4;

```

```
conversion_status_t status = big_to_little32((uint32_t *)source,test_length);
assert_int_equal(status, Invalid_pointer);
}
```

```
/*************************************************************
```

```
* * invalid_test_little_to_big() - This function is used to test NULL pointer
```

```
* * check in little_to_big32 function.
```

```
* * Input parameter - For this we send the NULL pointer as an input
```

```
* * parameter along with some length
```

```
* * Expected Result - Since an invalid(NULL) pointer is passed as a
```

```
* * parameter the function should return an Invalid_pointer enum value
```

```
* *
```

```
*****/
```

```
void invalid_test_little_to_big(void **state)
```

```
{
```

```
    uint32_t *source = NULL;
```

```
    uint32_t test_length = 4;
```

```
    conversion_status_t status = little_to_big32((uint32_t *)source,test_length);
```

```
    assert_int_equal(status, Invalid_pointer);
```

```
}
```

```
*****
```

```
* * valid_conversion_test_big_to_little() -This function is used to test a
```

```
* * valid conversion using big_to_little32 function.
```

```
* * Input parameter - For this we send a pointer to a valid data
```

```
* * along with the length in bytes
```

```
* * Expected Result - Since it does not have an invalid pointer or
```

```
* * invalid length it should successfully convert data and return Pass
```

```
* * enum value.
```

```

* *****/
void valid_conversion_test_big_to_little(void **state)
{
    uint32_t test_data = 0x12345678;
    uint32_t *ptr = &test_data;
    uint32_t length = 4;
    conversion_status_t status = big_to_little32(ptr,length);
    assert_int_equal(status, Pass);
}

/*****
 * * valid_conversion_test_little_to_big() - This function to test a
 * * valid conversion using little_to_big32 function.
 * * Input parameter - For this we send a pointer to a valid data
 * * along with the length in bytes
 * * Expected Result - Since it does not have an invalid pointer or
 * * invalid length it should successfully convert data and return Pass
 * * enum value.
 * *****/
void valid_conversion_test_little_to_big(void **state)
{
    uint32_t test_data = 0x12345678;
    uint32_t *source = &test_data;
    uint32_t length = 4;
    conversion_status_t status = little_to_big32(source,length);
    assert_int_equal(status, Pass);
}

int main(int argc, char **argv)

```

```

{
const struct CMUnitTest tests[] = {
    cmocka_unit_test(invalid_test_big_to_little),
    cmocka_unit_test(invalid_test_little_to_big),
    cmocka_unit_test(valid_conversion_test_big_to_little),
    cmocka_unit_test(valid_conversion_test_little_to_big),
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

*****
* * Authors : Harsimransingh Bindra
* * Date : 10/27/2017
* *
* * File : test_memory.c
* * Description : Source file for testing memory functions
* *      -testcircbuf_allocate_free()
* *      -testcircbuf_invalid_pointer()
* *      -testcircbuf_non_initialized_buffer()
* *      -testcircbuf_add_remove()
* *      -testcircbuf_buffer_full()
* *      -testcircbuf_buffer_empty()
* *      -testcircbuf_wrap_add()
* *      -testcircbuf_wrap_remove()
* *      -testcircbuf_over_fill()
* *      -testcircbuf_over_empty()
* *
* * Testing environment : To test the functions we need
* * to install the Cmocka framework libraries in our folder
* * path and make changes to the test Makefile to include

```

```

* * our source files and test files.

* *****/
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include "circbuf.h"
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>

/***/

* * Defining the Global variables required for out test function
* * buff_length gives length of buffer to access
* * test_data[6] array to behave as a buffer
* *****/
uint8_t buff_length = 6;
uint8_t test_data[6] = {1,2,3,4,5,6};

/***
* * testcircbuf_allocate_free() - This function is used to test CB_init
* * function and CB_Destroy function to check if buffer function can be used
* * to allocate and free memory from heap
* * Input parameter - For this we send the test_buff structure pointer.
* * to CB_init function and pass it again to the destory function
* * Expected Result - Receive a No_error enum from the functions
* *****/
void testcircbuf_allocate_free(void **state)
{

```

```

CB_t *test_buff = (CB_t *) calloc(1,sizeof(CB_t));

CB_init(test_buff, buff_length);

CB_status status = CB_destroy(test_buff);

assert_int_equal(status, No_error);

}

/**********************/

/* * testcircbuf_invalid_pointer() -This function is used to test if circular buffer function
 * can handle invalid pointer(NULL)

* * Input parameter - Here we send a NULL pointer to the
 * CB_is_full function

* * Expected Result - Receive a Null_pointer enum from the function
* *****/
void testcircbuf_invalid_pointer(void **state)
{
    CB_status status = CB_is_full(NULL);

    assert_int_equal(status,Null_pointer);

}

/**********************/

/* * noninitialized_circbuf_test() - This function is used to test if circular
 * buffer is initialized before using.

* * Input parameter - For this we allocate memory to the circular buffer
 * and send it to initialize function

* * Expected Result - Check if head and tail points towards the buffer and set
 * No_error enum
* *****/
void noninitialized_circbuf_test(void **state)

```

```

{
CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
CB_init(test_buffer, buff_length);
CB_status status;
if((test_buffer->head) == (test_buffer->buffer)) && ((test_buffer->tail) == (test_buffer->buffer)))
{
    status = No_error;
}
else status = Null_pointer;
assert_int_equal(status, No_error);
}

```

```

/********************************************************
* * testcircbuf_add_remove() - This function is used to test if circular buffer can
* * add and remove data to the buffer
* * Input parameter - For this we initialize our circular buffer and add data to the buffer
* * and then remove the data and check for the returned enum value
* * Expected Result - Receive a No_error enum from the functions
* *****/

```

```

void testcircbuf_add_remove(void **state)
{ uint8_t i=0;
CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
CB_init(test_buffer, buff_length);
CB_status status;
for(i=0; i<=5; i++)
{
    CB_buffer_add_item(test_buffer,i);
}
for(i=0; i<=5; i++)

```

```

{
    status = CB_buffer_remove_item(test_buffer);
}

assert_int_equal(status, No_error);

}

/**********************/

* * testcircbuf_buffer_full() - This function is used to test if circular buffer is
* * full and return corresponding enum value
* * Input parameter - For this we initialize the circular buffer and add data to it
* * until the circular buffer is full
* * Expected Result - Receive a Full enum from the function
* ****

void testcircbuf_buffer_full(void **state)
{
    uint8_t i=0;
    CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    for(i=0; i<11; i++)
    {
        CB_buffer_add_item(test_buffer,i);
    }
    CB_status status = CB_is_full(test_buffer);
    assert_int_equal(status, Full);
}

/**********************/

* * testcircbuf_buffer_empty() - This functionis used to test if circular buffer is
* * empty and return corresponding enum value

```

```

* * Input parameter - For this we initialize the circular buffer and add no data to it
* * but directly call our remove function
* * Expected Result - Since buffer is already empty we should receive a
* * Empty enum from the function
* *****/
void testcircbuf_buffer_empty(void **state)
{
    CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    CB_status status = CB_is_empty(test_buffer);
    assert_int_equal(status, Empty);
}

/***/

* * testcircbuf_wrap_add() - This Function is used to test if the circular buffer can wrap
* * near the boundary when data is added to the circular buffer.
* * Input parameter - For this we initialize buffer and add data to it
* * until buffer is full and check the head pointer with its previous value
* * Expected Result - If the head pointer and previous value point
* * at the same position then there is no error and No_error enum value should be set.
* *****/

```

```

void testcircbuf_wrap_add(void **state)
{
    uint8_t i=0;
    CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    uint8_t *last = (uint8_t *)(test_buffer->head);
    for(i=0;i<buff_length; i++)

```

```

{
    CB_buffer_add_item(test_buffer,i);

}

CB_status status;

if((test_buffer->head)==last)
    status = No_error;
else status = Error;

assert_int_equal(status, No_error);

}

/*****************/
/* * testcircbuf_wrap_remove() - This function is used to test if circular buffer can
 * * wrap near the boundary when data is removed from it.
 * * Input parameter - For this we initialize buffer and add data to it
 * * until buffer is full and remove the same amount of data then check the
 * * tail pointer with its previous value.
 * * Expected Result - If the tail pointer and previous value point
 * * at the same position then there is no error and No_error enum value should be set
 * */
void testcircbuf_wrap_remove(void **state)
{
    uint8_t i=0;

    CB_t *test_buffer = (CB_t *) calloc(1,sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    uint8_t *last = (uint8_t * )(test_buffer->tail);
    for(i=0; i<buff_length; i++)
    {
        CB_buffer_add_item(test_buffer,i);
    }
}

```

```

}

for(i=0;i<buff_length;i++)
{
    CB_buffer_remove_item(test_buffer);
}

CB_status status;
if((test_buffer->tail)==last)
    status = No_error;
else status = Error;

assert_int_equal(status, No_error);

}

/*****************/
/* * testcircbuf_over_fill() - This function is used to test if the circular buffer can
 * handle when more data is added than its size
 * Input parameter - For this we initialize buffer and add data more
 * than its capacity
 * Expected Result - Once the buffer is full if anymore data is added to
 * it, it will return a Full enum from the function
* *****/

```

```

void testcircbuf_over_fill(void **state)
{
    uint8_t i=0;
    CB_t *test_buffer = (CB_t *) malloc(sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    CB_status status;
    for(i=0; i<=buff_length; i++)

```

```

{
    status = CB_buffer_add_item(test_buffer,i);
}
assert_int_equal(status, Full);
}

/*****************/
/* * testcircbuf_over_empty() - This function is used to test if circular buffer can
 * handle when more data is removed than available in the buffer
 * Input parameter - For this we initialize buffer and call our remove function
 * Expected Result - Since the buffer is already empty if we remove anymore
 * data from it it will return a Empty enum from the function
 */
void testcircbuf_over_empty(void **state)
{
    CB_t *test_buffer = (CB_t *) malloc(sizeof(CB_t));
    CB_init(test_buffer, buff_length);
    CB_status status = CB_buffer_remove_item(test_buffer);
    assert_int_equal(status, Empty);
}

int main(int argc, char **argv)
{
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(testcircbuf_allocate_free),
        cmocka_unit_test(testcircbuf_invalid_pointer),
        cmocka_unit_test(noninitialized_circbuf_test),
        cmocka_unit_test(testcircbuf_add_remove),
        cmocka_unit_test(testcircbuf_buffer_full),
    }
}

```

```

cmocka_unit_test(testcircbuf_buffer_empty),
cmocka_unit_test(testcircbuf_wrap_add),
cmocka_unit_test(testcircbuf_wrap_remove),
cmocka_unit_test(testcircbuf_over_fill),
cmocka_unit_test(testcircbuf_over_empty),
};

return cmocka_run_group_tests(tests, NULL, NULL);
}

*****
* * Authors: Harsimransingh Bindra, Smitesh Modak
* * Date Edited: 10/01/2017
* *
* * File: conversion.c
* *
* * Description: source file for data manipulation
* * -my_itoa
* * -my_atoi
* * -big_to_little32
* * -little_to_big32
* *
* *
*****/

#include <stdint.h>
#include <stdio.h>
#include "conversion.h"           //invokes user defined header file for conversion
uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base) //function to convert integers to ascii strings
{
    uint8_t rem, i, temp, digits, count = 0;

```

```

uint8_t sign = 0;

uint8_t *last;

if(data<0) //checks if data is negative and appends - to the string

{
    data = -data;

    *ptr = '-';
    ptr++;
    count++;
    sign++; //to check the sign of
the input

}

if(data==0) //to return zero
value if zero is entered

    *ptr='0';
    return 0;
}

while(data != 0)

{
    if((rem = data%base) > 9) //condition if there is hex conversion
    {

        *ptr=('A'+ rem-10); //adds the ascii value of 'A' character to
remainder

    }

    else

    {
        *ptr=rem + '0'; //for normal conversion adds the ascii
value of '0' to the remainder
    }

    data = data/base;
}

```

```

        ptr++;
        count++;
    }

*ptr = '\0';

last = ptr - sizeof(uint8_t);

digits = count;

if(sign == 1)

{
    ptr = ptr - sizeof(uint8_t)*(count-1);

}

else

{
    ptr = ptr - sizeof(uint8_t)*(count);

}

if(count%2 != 0)           //to check the string length whether its even or odd

{
    count = count-1;
}

for(i=0; i<(count/2); i++)      //to reverse the string and store the ascii string in correct
order

{
    temp = *last;
    *last = *ptr;
    *ptr = temp;
    ptr++;
    last--;
}

ptr = ptr - sizeof(uint8_t)*(count/2);

return digits;

```

```

}

int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base)//function to convert ascii strings to integers
{
    if(ptr == NULL)
    {
        return Invalid_pointer;
    }

    int32_t integer=0;
    uint8_t i, sign=0;

    if(*ptr == '-')
        //to check whether the first character is '-' and set the sign
bit accordingly
    {

        sign++;

        ptr++;
    }

    digits = digits-1;

    }

    for(i = 0; i<digits; i++)           //to convert ascii strings to integers
    {

        if('0' <= *ptr && *ptr <='9')

        {

            integer = integer*base + *ptr - '0';           //subtracts the
            ptr++;
        }

        else

        {

            integer = integer*base + *ptr -55;
            ptr++;
        }
    }
}

```

```

    }

    if (sign == 1)          // to assign the proper sign to the integer according to the
sign value

    {

        integer = -integer;

    }

    return integer;

}

int8_t big_to_little32(uint32_t *data, uint32_t length) //to change the endianness from big to little
{
    if(data == NULL)

    {

        return Invalid_pointer;

    }

    uint32_t temp, temp1, temp2, temp3;

    temp = *data & 0xFF000000;           //masking all but msb

    temp1 = *data & 0X000000FF;

    temp2 = *data & 0X00FF0000;

    temp3 = *data & 0X0000FF00;

    temp = temp >> 24;                //shifting msb to lsb

    temp1 = temp1<<24;

    temp2 = temp2>>8;

    temp3 = temp3<<8;

    *data = temp | temp1 | temp2 | temp3;      //assembles all bytes after proper shifting
and getting the required endianness

    return Pass;

}

```

```

int8_t little_to_big32(uint32_t *data, uint32_t length) //to change the endianness from little to big
{
    if(data == NULL)
    {
        return Invalid_pointer;
    }

    uint32_t temp, temp1, temp2, temp3;

    temp = *data & 0XFF000000;           //masking the lsb
    temp1 = *data & 0X000000FF;
    temp2 = *data & 0X00FF0000;
    temp3 = *data & 0X0000FF00;
    temp = temp >> 24;                //shifting the lsb
    temp1 = temp1<<24;
    temp2 = temp2>>8;
    temp3 = temp3<<8;

    *data = temp | temp1 | temp2 | temp3; //assembles all the bytes after proper
shift to get the required endianness

    return Pass;
}

```

```

*****
* * Authors: Harsimransingh Bindra, Smitesh Modak
* * Date Edited: 10/01/2017
* *
* * File: memory.c
* *
* * Description: source file for data manipulation
* * -my_memmove
* * -my_memcpy

```

```

* * -my_memset
* * -my_memzero
* * -my_reverse
* * -reserve_words
* * -free_words
*
*
* ****

```

```

#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "memory.h"

uint8_t my_memmove(uint8_t *src, uint8_t *dst, size_t length) //function to move data from one
memory location to another
{
    if(src == NULL || dst == NULL)
    {
        return Invalid_pointer;
    }
    uint8_t i = 0;
    if((src+sizeof(uint8_t)*length)>=dst && src<dst)      //Checks for overlapping of memory location
    {
        src = src+(sizeof(uint8_t)*(length-1));
        dst = dst+(sizeof(uint8_t)*(length-1));
        for(i = 0; i<length; i++)                      //to move data to the destination memory location
        {
            *dst = *src;

```

```

        src--;
        dst--;
    }

    return Pass;
}

else
{
    for(i = 0; i<length; i++)
    {
        *dst = *src;

        src++;
        dst++;
    }

    return Pass;
}

uint8_t my_memcpy(uint8_t *src, uint8_t *dst, size_t length) //function to copy data from one memory
location to another

{
    if(src == NULL || dst == NULL)
    {
        return Invalid_pointer;
    }

    uint8_t i = 0;

    for(i = 0; i<length; i++)           //to copy data from source memory address to destination
memory location

    {
        *dst = *src;
    }
}

```

```

        src++;
        dst++;
    }

    return Pass;
}

```

uint8_t my_memset(uint8_t *src, size_t length, uint8_t value)//to set all given memory locations with a particular value

```

{
    if(src == NULL)
    {
        return Invalid_pointer;
    }
    size_t i = length;
    while(i != 0)           //to set a particular value
    {

```

```

        *src = value;
        i--;
        src++;
    }

```

src = src-(sizeof(uint8_t)*length);

return Pass;

}

uint8_t my_memzero(uint8_t *src, size_t length) //function to set all values at given memory locations to zero

```
{
    if(src == NULL)
```

```

{
    return Invalid_pointer;
}

size_t i = length;

    while(i != 0)

    {
        *src = 0;                                //sets all values to zero

        i--;
        src++;
    }

    src = src-(sizeof(uint8_t)*length);

    return Pass;
}

```

```

uint8_t my_reverse(uint8_t *src, size_t length)      //function to reverse a string

{
    if(src == NULL)

    {
        return Invalid_pointer;
    }

    uint8_t *last;

    uint8_t temp,i;

    last = src + sizeof(uint8_t)*length-1;

    if(length%2 != 0)                          //to check whether the length of the string or number of
values are even or odd

    {
        length = length-1;
    }

    for(i = 0; i<(length/2); i++)           //to reverse the string at given memory location

```

```

    {
        temp = *src;
        *src = *last;
        *last = temp;
        src++;
        last--;
    }

    src = src - (sizeof(uint8_t)*(length/2));
    return Pass;
}

uint32_t *reserve_words(size_t length)
{
    uint32_t *reserve = malloc(sizeof(uint32_t)*length);
    return reserve;
}

void free_words(uint32_t *src) //function to free the dynamically allocated memory
{
    free(src);
}

```